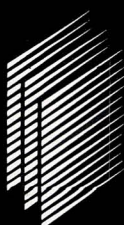


# PROFESSIONAL DRAW 3.0

## Supplement Manual



*GOLD DISK*



# **PROFESSIONAL DRAW**

*Version 3.0*

---

Supplement Manual

---

All products mentioned in this manual are trademarks of their respective owners.

Professional Page and Professional Draw are trademarks of Gold Disk Inc.

Gold Disk and the Gold Disk logo are registered trademarks of Gold Disk Inc.

PANTONE®\* Computer Video simulations used in this product may not match PANTONE-identified solid color standards. Use current PANTONE Color Reference Manuals for accurate color.

Pantone, Inc. is the copyright owner of PANTONE Color Computer Graphics and Software which are licensed to Gold Disk Inc. to distribute for use only in combination with Professional Draw. PANTONE Color Computer Graphics and Software shall not be copied onto another diskette nor into memory unless as part of the execution of Professional Draw.

"PANTONE Color Computer Graphics," © Pantone, Inc., 1986, 1988.

©1992 Gold Disk Inc. All rights reserved.

No part of this manual may be copied or distributed, transmitted, transcribed, stored in a retrieval system, or translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, manual, or otherwise, without the express written permission of Gold Disk Inc.

Printed In Canada

Gold Disk Inc.  
P.O. Box 789, Streetsville  
Mississauga, Ontario  
Canada L5M 2C2

\* Pantone, Inc.'s check-standard trademark for color reproduction and color reproduction materials.

# Table of Contents

|                                          |          |
|------------------------------------------|----------|
| <b>1. Introduction</b>                   | <b>1</b> |
| <b>2. Hard Disk Installation</b>         | <b>3</b> |
| <b>3. New Features in 3.0</b>            | <b>5</b> |
| The New Requesters                       | 5        |
| Hot-link to Professional Page            | 5        |
| Bitmap Graphics and EPSF Files           | 6        |
| Positioning                              | 6        |
| Sizing                                   | 6        |
| Cropping                                 | 7        |
| Rotating                                 | 7        |
| Hints for Scanning and Importing Bitmaps | 8        |
| Color and Fills                          | 8        |
| The PANTONE Colors                       | 8        |
| Gradient Fills                           | 9        |
| Clearing Unused Colors from the Palette  | 10       |
| 24 Bit Color Support                     | 10       |
| Text Improvements                        | 10       |
| Compugraphic Font Support                | 10       |
| Adobe Type 1 Font Support                | 11       |
| Text Requester                           | 13       |
| Text Placing and Editing                 | 13       |
| Viewing Artwork                          | 14       |
| Object/Outline                           | 14       |
| Object/WYSIWYG                           | 14       |
| Autotrace Improvements                   | 14       |
| Improved Color Separation                | 15       |
| Autotiling Output                        | 15       |
| Dot Matrix Offset                        | 16       |
| HPGL Scaling and Offset                  | 16       |
| Undo                                     | 17       |
| Redo                                     | 17       |

|                                       |           |
|---------------------------------------|-----------|
| <b>4. Using Professional Draw 3.0</b> | <b>19</b> |
| Text on a Curve                       | 19        |
| Blend                                 | 19        |
| Add Control Point Tool                | 19        |
| Cut Control Point Tool                | 20        |
| Joining Line Segments                 | 20        |
| The Help Function                     | 21        |
| HPGL Plotter Support                  | 21        |
| Saving Clips                          | 21        |
| Saving Clips from Professional Draw   | 21        |
| Saving Clips from Trace               | 21        |
| <b>5. Function and Tool Genies</b>    | <b>23</b> |
| Executing a Function Genie            | 23        |
| Using the About Button                | 23        |
| Defining and Modifying Genies         | 24        |
| Function Genie List Management        | 24        |
| Function Genie Keys                   | 25        |
| Unexpected Returns                    | 25        |
| The Function Genies                   | 25        |
| AddToLabelDataBase                    | 25        |
| AdjustColors                          | 26        |
| ADProHotLink                          | 26        |
| AutoSave                              | 26        |
| AveryLabels                           | 27        |
| ClickRadFill                          | 27        |
| Command                               | 27        |
| ConvertToBW                           | 28        |
| ConvertToGray                         | 28        |
| CopyObjectAttr                        | 28        |
| CopyObjToPages                        | 28        |
| CopyPages                             | 28        |
| DeleteRange                           | 28        |
| DistributeObject                      | 29        |
| ObjectToLayer                         | 29        |
| PointsAlign                           | 29        |
| PointsMove                            | 29        |

|                                      |           |
|--------------------------------------|-----------|
| PointsOffset                         | 30        |
| SavePrefs                            | 30        |
| ScaleToSize                          | 30        |
| SpaceObjects                         | 30        |
| SpacePoints                          | 30        |
| StepAndRepeat                        | 30        |
| StyleTagAdd                          | 31        |
| StyleTagApply                        | 31        |
| StyleTagDelete                       | 31        |
| TileSelection                        | 31        |
| UnitsConverter                       | 31        |
| Flower                               | 32        |
| Mandela                              | 32        |
| PlotFunction                         | 32        |
| PlotPolar                            | 32        |
| Tool Genies                          | 33        |
| Executing a Tool Genie               | 33        |
| Marquee Tool                         | 33        |
| Rectangle Tool                       | 33        |
| Ellipse Tool                         | 33        |
| Pen Tool                             | 33        |
| Creating and Altering Tool Genies    | 33        |
| <b>6. ARexx Commands and Support</b> | <b>35</b> |
| Technical Information                | 35        |
| Parameter and Return Information     | 36        |
| Page and Object Identifiers          | 36        |
| General Functions                    | 37        |
| User Interactive Functions           | 38        |
| Project Menu Commands                | 40        |
| Page Operations                      | 41        |
| Object Operations                    | 42        |
| Object Attribute Operations          | 46        |
| Clip Operations                      | 49        |
| Environment Operations               | 50        |
| Color Palette Operations             | 51        |
| Drawing Operations                   | 52        |

|                              |           |
|------------------------------|-----------|
| Text Operations              | 53        |
| Control Point Manipulation   | 54        |
| Page PostScript Output Specs | 55        |
| PostScript Printing          | 56        |
| Thumbnail Printing           | 59        |
| <b>7. The Genie Editor</b>   | <b>61</b> |
| To Use the Genie Editor      | 61        |
| Entering Text                | 61        |
| Project Menu                 | 61        |
| Edit Menu                    | 62        |
| Find Menu                    | 62        |



# 1 ■ Introduction

This addendum explains features added to **Professional Draw 3.0**. The basics of **Professional Draw** are the same, but the general look has been updated to conform with *Workbench 2.0* and *Professional Page 3.0*. **Professional Draw 3.0** also contains:

- Advanced bitmap and EPSF positioning, sizing and rotation
- PANTONE®\* color support
- Gradient color fills
- 24 bit color support
- Direct Compugraphic Font support
- Adobe Type 1 Font support
- Improved text handling
- New WYSIWYG and Outline viewing commands
- Improved autotracing
- Autotiling
- Undo and Redo commands
- AREXX support
- Function and Tool Genies
- An Editor for creating and modifying ARexx scripts and Genies

This addendum describes the new look to **Professional Draw 3.0**, and explains the new features. It also contains hints designed to make it easier to use some features of **Professional Draw**. Changes and additions are usually cross referenced with the **Professional Draw 2.0** manual, but you may wish to cross-reference or write the changes in your manual.

\* Pantone, Inc.'s check-standard trademark for color reproduction and color reproduction materials.



## 2. Hard Disk Installation

To use version 3.0 of **Professional Draw**, it must be installed on your hard drive. Installation is accomplished by running *Install\_PDraw*. **You must run this program from the Workbench!**

### To install **Professional Draw**

- ➡ Insert disk 1 of **Professional Draw 3.0** into your disk drive, double-click on its icon to open its window, then double-click on the *Install\_PDraw* icon within the window.

If you already have **Professional Draw** installed, you will be asked if you want to update it to the new version.

If you do not already have **Professional Draw** installed, a requester will appear asking where you want to install it. In this requester, you can select from your different hard drive partitions and directories, and you can create a new directory if you so desire. It is recommended that you install **Professional Draw** into its own directory, named "*PDraw*".

- ➡ Select the directory in which you want the **Professional Draw** programs and data installed.

If you already have *CGFonts*: assigned, **Professional Draw**'s *CGFonts* support files will be installed in that directory.

If you do not have *CGFonts*: assigned, a *CGFonts* directory will be created in your installation directory, and an *assign* command for *CGFonts*: will be added to your startup-sequence.

If you already have *rexx*: assigned, the **Professional Draw Genies** will be installed into the *rexx* directory.

If you do not have *rexx*: assigned, you will be asked if you want to use any Genies with **Professional Draw**.

- ➡ To use Genies, select *Yes*, and then select the drawer where you want the genies to be located.

If you already have **Professional Draw** fonts on your system (if *PDrawFonts*: is already assigned), they will be automatically used by **Professional Draw 3.0**.

If you do not already have **Professional Draw** fonts, they will be installed into a *PDrawFonts* directory within your installation directory.

Finally, you will be asked if you want *FontManager* (to convert Adobe Type 1 fonts to CGFonts), the new version of *Trace*, *ClipMap* (to convert *Clips* to IFF bitmaps), and the structured clipart library.

➡ Click on the items you wish to install (so that there is a checkmark next to them), then click on *Proceed*.

The items you have indicated will be installed into your **Professional Draw** directory. In the case of the structured clipart, it will be installed into a directory named "*Clips*".

# 3. New Features in 3.0

## The New Requesters

The requesters in *Professional Draw* have been changed to reflect the look of *Workbench 2.0* and *Professional Page 3.0*.

The requesters contain the same information and options as they did in version 2.0, but the presentation is slightly different. For example, the *Disk Drive* and *Partition* buttons have been given a 3-D appearance.

You can now scroll through *List* windows using the scroll bar, or the *Up* and *Down* arrows located beneath the scroll bar.

The most important change to the file requester is that the files are now automatically sorted alphabetically, with drawers and directories appearing at the top of the list.

See Chapter 4, "Basic Concepts," (page 44) of the *Professional Draw 2.0* manual for more information on using requesters.

## Hot-Link to *Professional Page*

Users of *Professional Page* may now access the full power of *Professional Draw* directly from *Professional Page*. Clips may be transferred from *Professional Page* to *Professional Draw* for editing or adjustment. For the hot-link to work, *Professional Draw* and *Professional Page* must be running. This will only work with *Professional Draw 3.0* and *Professional Page 3.0* or higher.

To operate the hot-link from *Professional Page*, simply click on a box containing a *Professional Draw* clip and select *PDraw* from the *Draw* menu or press Amiga-backslash (the Amiga key to the right of the space bar and the slash on the keyboard's top row).

The *Professional Draw* screen will be brought to the front, with the clip on a new blank page. Once you have finished editing the clip, select all of the objects you want the new clip to contain, and select *Send Clip Home* from the *Special* menu. The clip will be automatically re-imported into your *Professional Page* document.

# Bitmap Graphics and EPSF Files

*Professional Draw* will import any standard IFF compatible image, including HAM (Hold and Modify) images. (Chapter 9, *Bitmaps*, in the *Professional Draw 2.0* manual covers working with Bitmap images.) *Professional Draw* can also import EPSF files, but will not display them on the screen. You will see a gray box instead.

*Professional Draw 3.0* includes enhancements to positioning, sizing and cropping Bitmaps and EPSF files.

## Positioning

A bitmap or EPSF image can be moved anywhere on the drawing page or the drawing board by dragging the image, or by entering the co-ordinates in the *Object Position* requester.

### To drag the image:

- Use the *Null Pointer* tool and click anywhere on the bitmap or EPSF, hold the mouse button down, and drag the image to the desired position.

### To enter co-ordinates in the requester:

- Select the bitmap or EPSF using the *Null Pointer*, then double click on the *Null Pointer* tool.

The *Object Position* requester will appear on the screen.

- Enter the desired co-ordinates for the upper left corner of the bounding box.
- Click on *OK*.

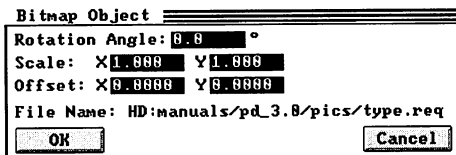
The image will be placed at the new co-ordinates.

See pages 69 and 70 in the *Professional Draw 2.0* manual for more information on moving objects.

## Sizing

*Professional Draw 2.0* allows you to resize bitmaps and EPSFs using the mouse and *Alt*-key combinations (Page 96, *Professional Draw 2.0* manual). Version 3.0 includes new *Bitmap Object* and *EPSF Object* requesters which contain options for setting X

and Y axes scale and offset values. The *EPSF Object* requester also contains information about the size and location of the file.



### To change the size of an image:

- Double click on the image.

The *Bitmap Object* or *EPSF Object* requester will appear.

- Enter the new scale values (these values will be remembered).

Values less than 1.0 will reduce the object's size, values greater than 1.0 will enlarge it. Using different values for X and Y allows for distortion, and amorphic scaling.

Bitmaps cannot be flipped by using negative scaling values like structured graphics can be.

## Cropping

The offset option in the *Bitmap Object* and *EPSF Object* requesters allows you to offset the image within its bounding box. You can even crop an image by moving part of the image outside the bounding box so that it will not be displayed or printed. However, it may be difficult to crop EPSF images as the actual image does not appear on your screen.

### To crop an image:

- Double click on the image to bring up the *Bitmap Object* or *EPSF Object* requester.

- Enter new offset values.

Offset values are specified as distances from the top left corner of the bounding box.

See page 96 of the *Professional Draw 2.0* manual for more information on cropping images.

## Rotating

*Professional Draw 3.0* allows you to rotate a bitmap or EPSF image using the *Rotation* tool.

### To rotate an image:

- With the *Null Pointer*, select the bitmap or EPSF to be rotated.
- Click on the *Rotation* tool.
- Move the crosshairs to the desired rotation axis and click the left mouse button.
- Move the crosshairs to another point, click and hold the left mouse button. Drag the image to its new position.

You can also rotate images numerically by double clicking on the rotation tool to bring up the *Bitmap Object* or *EPSF Object* requester.

See page 73 of the *Professional Draw 2.0* manual for more information on *Rotation*.

## Hints for Scanning and Importing Bitmaps

The size and aspect ratio of an image affects the imported image. Pages scanned at 300 DPI, or pictures that include all of the white space (blank area) surrounding the actual image, may be too large for your monitor to display all at once.

If the whole image, including the white space, is imported into *Professional Draw*, it may be difficult to get the desired portion of the image onto *Professional Draw*'s working page.

Paint programs such as *Deluxe Paint* from Electronic Arts will scroll around the page so that different parts of the image are visible on your screen. Using such software, you can save the necessary portion of a scanned image as a brush. This brush can be imported into *Professional Draw* and handled more easily than larger bitmaps.

## Color and Fills

Several enhancements have been made to the *Color* and *Fill* options in *Professional Draw 3.0*. These are: the addition of a video simulation of the PANTONE MATCHING SYSTEM®, gradient fills for structured objects, a *Clean Palette* command, and 24 bit color support.

### The PANTONE Colors

The video simulation of PANTONE colors includes a database of more than 700 special colors along with the four process colors. This is particularly important to designers who regularly specify these colors and need close approximation in color compositions and presentations for output on color PostScript devices.

The PANTONE MATCHING SYSTEM has been developed to provide designers and printers with a standard for color matching based on precise ink mixing formulas. We recommend use of one of the many *PANTONE Library of Color* reference guides, such as the *Color Formula Guide* or the *Color Specifier*. These are available from graphic art suppliers and paper merchants.

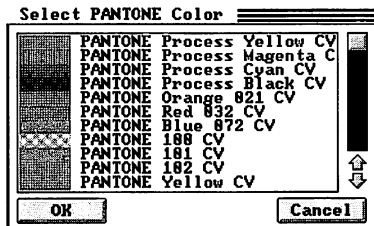
The colors displayed by your monitor may not match your output. The dithering methods employed by *Professional Draw*, provide screen approximations of the actual color. Duplication of the PANTONE Colors using the 4-process colors - cyan, magenta, yellow, and black (CMYK) - requires special combinations of film tint densities. When printing on a color PostScript device, *Professional Draw* will simulate the PANTONE Colors as closely as possible using the standard process colors.

\* Pantone, Inc.'s check-standard trademark for color reproduction and color reproduction materials.



**PANTONE colors cannot be edited** but their intensity, or tint, can be modified by adding a percentage value to the color name. To do this:

- ➡ Select the object you want to fill.
- ➡ Select *Attributes/Fill/Solid*.
- ➡ Click on the *Palette* gadget.



- ➡ Click on the *PANTONE* gadget and choose a color from the database.
- ➡ Click after the color's name in the text gadget, leave a space, and type a percentage value for the degree of intensity you require. Lower percentages create lighter tints.

Notice that the CMYK values have been modified to reflect the new color intensity. PANTONE Colors that have been modified are automatically named to reflect the new intensity percentage of the tint.

## Gradient Fills

*Gradient* is a new function in the *Attributes/Fill* Menu. There are two types of definable gradient fills - *radial* (similar to a circular blend), and *linear*. Both can be styled by color, and number of steps. Radial gradient fills have a defined center and linear gradient fills can be created on any angle.

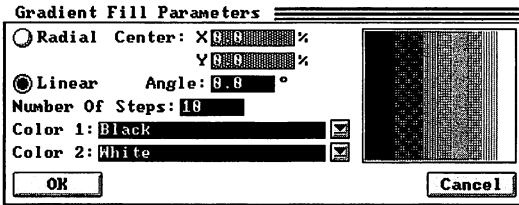
The center of a radial gradient fill can be placed by specifying a percentage value for the distance from the center of the object's bounding box. For example, to place the center of the radial gradient at the center of the bounding box, you would type 0 for both the X and Y axes. Moving the center to 100% on both the X and Y axes would place the gradient center in the lower right corner, and a center of -100% on both axes would place the gradient center in the upper left corner of the bounding box.

**NOTE:** Gradient fill positioning is based on the edge of the object's bounding box, not the edge of the object itself.

Gradient fills can be used on any object or text, easily creating many effects that are similar to blends. The tutorial on pages 25 to 27 of the *Professional Draw 2.0* manual describes a method for creating a airbrush like blend of colors on the letters of a word. The method describes creating a mask and compound object.

**Here is the same effect using a gradient fill:**

- ➡ Using the *Text* tool type AMIGA using a point size of 48.
- ➡ Select *Attributes/Fill/Gradient*.



- ➔ Select colors for both Color 1 and Color 2.
- ➔ In the requester, change the number of steps and choose the linear type of fill.

Each step is one shade of the graduation between Color 1 and Color 2. Therefore the more steps you specify, the more gradual the transition will be.

- ➔ Click on the *OK* gadget.

Notice that each letter has a gradient. If you want the gradient to graduate from light to dark across the whole word:

- ➔ Use the *Null Pointer Tool* and select the word.
- ➔ Select *Objects Menu/Make Compound Object*.

Now the gradient fill goes from light to dark across the whole word.

## Clearing Unused Colors from the Palette

The *Clean Palette* command has been added to the *Special* menu. Selecting this command removes any colors from the palette that are not used in the current document or clip list.

## 24 Bit Color Support

24 bit color support has been added to version 3.0. *Professional Draw* will now load 24 bit images in the same manner as any other image. For more information on loading files in general, see page 17 of the *Professional Draw 2.0* manual.

## Text Improvements

Several text handling improvements have been made in *Professional Draw 3.0*. These include direct CGFont support, the ability to set line spacing, bolding for CGFonts and the aspect ratio. You can also edit text on-screen, and group text so that you can justify it.

## Compugraphic Font Support

*Professional Draw 3.0* directly supports Compugraphic Fonts. You no longer have to convert Compugraphic Fonts to PDraw fonts using *CreateFont*; *Professional Draw* will read them directly. PDraw fonts are displayed in the Text Requester with a "(PD)" after the typeface name to differentiate the font types.

*Professional Draw* can use CGFonts available from *Gold Disk* and standard Workbench 2.0 Compugraphic fonts. If you install more fonts onto Workbench 2.0 after you have installed *Professional Draw*, you will have to run the *CGUpdate*

program in order for **Professional Draw** to recognize the new fonts. *CGUpdate* can be found in the same drawer as **Professional Draw**.

**Note:** You only need to run *CGUpdate* once each time you install new Compugraphic fonts into Workbench 2.0.

## Adobe Type 1 Font Support

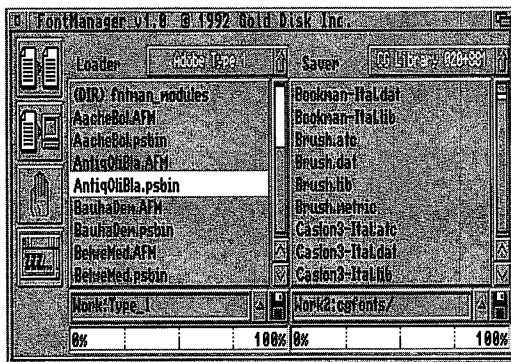
A new utility has been added to the **Professional Draw** package. This utility, *FontManager*, is used to convert an Adobe Type 1 font to a Compugraphic format font. Users of **Professional Draw** may now access the full Adobe type library. In addition, the Compugraphic output of *FontManager* may also be used as system fonts under Workbench 2.0.

### Using *FontManager*

To convert an Adobe Type 1 font into a Compugraphic font, two files are required: the Printer Font Binary file and the Adobe Font metric file. The Printer Font Binary file contains the actual character outlines and, when supplied on MS-DOS format disks, these files are generally named <fontname>.PFB. The Adobe Font Metric file contains information concerning the placement of characters; i.e., width and kerning data. These files are generally named <fontname>.AFM on MS-DOS format disks. Both of these files must be present in *FontManager*'s source directory for each Type 1 that you want to convert.

If you acquire Adobe fonts in a format other than MS-DOS, your font filenames may use other extensions. In that case, you will need to check with your supplier to determine which file contains font outlines and which file contains font metrics. *FontManager* will transparently convert fonts from any source - MS-DOS, Macintosh or Amiga.

Run *FontManager* by double-clicking on its icon. The program opens a window with a dual-directory display, one for source directory (where the font binary and font metric



files are located) and one for destination directory (generally *CGFonts*:). Each directory list box has the standard gadgets associated with such objects; the scrollbar, a pair of scroll buttons and a string gadget containing the pathname. In addition, each list box has four other gadgets.

The first is the *module* gadget which is located atop the

directory list. The source module is *Adobe Type 1* and the destination module is either *CG Library* or *CG Library 020+881*. If you are using an accelerated Amiga, you will want to use the latter module since a significant increase in speed will be realized. The second and third gadgets (the disk icon and the up-arrow by its side) offer an alternative to typing the pathnames into the string gadgets. Clicking on the disk icon will put a device list into the list box. This list contains physical devices (indicated as <DSK>) as well as assigned volumes (indicated as <LOG>). Double-click the name of the device and the files and subdirectories of that device will be displayed.

Double-click on the directory that contains your Type 1 fonts. If you need to back up, click on the up-arrow gadget to view the parent directory. When the filenames of your Type 1 fonts appear in the requester, click on the font binary file of the font that you wish to convert. If you wish, you may shift-click to select multiple fonts for conversion.

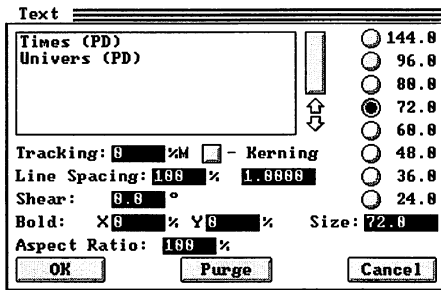
The fourth gadget is a 'fuel gauge' that is used to show the status of read/write operations. The left gadget will fill as the source font is read in. When that gauge reaches 100%, the destination gadget will begin to fill. When this gauge has reached 100%, the conversion is complete.

At the left of the window are the four gadgets that drive the application. Top to bottom, the four icons represent: *Convert*, *Preview*, *Abort* and *Iconify*. *Convert* will read in the source files and write out the proper Compugraphic format files to the destination directory. *Preview* will read in the source files and display a sample of the typeface in a new window. *Abort* will terminate an operation after first requesting your confirmation. If you're running Workbench 2.0, clicking *Iconify* will collapse *FontManager*'s window into a tiny window.

To be used with **Professional Draw**, each Compugraphic font must be comprised of the following files: <fontname>.dat and <fontname>.lib. With these files in your *CGFonts*: volume, the font is ready to be used in your documents. These files are all that is needed for the onscreen display and for printing.

When you click on *Convert*, *FontManager* will ask you if you want a downloadable font file to be placed into the *CGFonts:ps* directory. If you only wish to use this font with **Professional Draw**, you should always answer *No* to this prompt. **Professional Draw** does not need to use PostScript downloadable fonts. If you plan to use the font with *Professional Page*, you may wish to answer affirmatively. Consult your *Professional Page* documentation for more information on using *FontManager* with *Professional Page*.

## Text Requester



The Text Requester appears when you select the *Text* tool. It contains the usual typeface selection, kerning and tracking et cetera, but it also contains options to set line spacing, X/Y bolding for Compugraphic Fonts, shearing, and the aspect ratio. Once you have selected your settings, click on *OK*. You can now type your text directly onto the page.

### Line Spacing

Line spacing can either be set as a percentage of the current font size, or as a distance. Simply enter the percentage or the distance you want to use in the appropriate gadget. Distances are measured in whatever units you are currently using in the document.

### X/Y Bolding

X and Y bolding of Compugraphic fonts is set using a relative number. 0 being the normal font, 100 approximating a normal bolded font, and -100 approximating a light font. The X value will set the horizontal stroke thickness, and the Y value will set the vertical stroke thickness. Altering these values will not change the overall size of the font. X and Y bolding will have no effect on PDraw fonts.

### Shearing.

To shear text, simply enter the degree angle you want to shear the text on, in the shear gadget.

### Aspect Ratio

The aspect ratio is set as the percentage ratio of a character's width to its height. If you enter 75% for the aspect ratio, characters will be 3/4 as wide as they are tall.

## Text Placing and Editing

Once you have closed the Text Requester, you can begin adding text to your document. Simply click where you want the text to begin. A text cursor will appear and you can begin typing.

### Extended Character Set

Characters from the extended character set can be entered in the normal Commodore mode, or by holding down the [ALT] key, and entering the ASCII value on your keyboard's numeric keypad.

## Editing

To edit text, simply use the cursor keys to move through the text, and the [DELETE] and [BACKSPACE] keys to edit.

You can place the cursor within existing text by clicking on a letter in the text with the *Text* tool selected. A cursor will appear at the selected letter.

When you add text to your document it is placed in a single block or group. To justify this text, you must break the group into single lines of text.

## Justifying Text

There is a new menu item, *Re-Group by line* in the *Special* menu, that allows you to split a group of text into individual lines of text. You can then justify these lines using *Object/Align/ left or right* or *Object/Center/Vertical*.

## Viewing Artwork

Two new commands have been added to the *Object* menu that allow you to view some objects in WYSIWYG format, and others in outline format. This will allow you to see the full appearance of objects you are working on, while displaying all other objects in outline to speed up the screen redraw. These commands will have no visual effect if you are in Wireframe mode.

### Object/Outline

The *Outline* command in the *Object* menu cause the currently selected objects to be displayed as outlines, whether or not *Professional Draw* is in Wireframe mode. This may help speed up operations by cutting down on screen redraw times.

### Object/WYSIWYG

The *WYSIWYG* command in the *Object* menu will cause any object in Wireframe mode, to be viewed in full WYSIWYG mode.

## Autotrace Improvements

Autotrace can now trace bitmaps of any size and is no longer restricted to bitmaps smaller than 1024 X 1024.

*Trace* also has improved contour generation. This means that your traced image will more accurately reflect the original.

In version 3.0, *Trace* still works best on simple bitmaps with few colors. It is ideal for tracing logos or pictures created in a paint program, but may have difficulty tracing complex bitmaps such as scanned photographs because photographs will contain too many colors for *Professional Draw* to handle. It will not trace 24 bit bitmaps.

To improve the quality of your traces:

- Use simple bitmaps with a few distinct colors.
- Modify complex pictures using an image processor program (such as *Art Department Professional* from ASDG) to scale your bitmap and/or reduce the number of colors.
- Try tracing the bitmap a few times, changing the *Fit* parameter each time. Larger *Fit* parameter values produce smoother but less accurate curves.
- Perfect your traced clip within *Professional Draw* by modifying the control points and tangent lines.

## Improved Color Separation

Several improvements have been made to the color separation functions. Of primary concern to the user is the new manner of defining Undercolor Removal (UCR). UCR may now be specified as a threshold (expressed as a percentage). The percentage that you enter in the output requester specifies the minimum of YMC (yellow, magenta and cyan) below which *no* color is removed. Thus UCR now removes gray only from the darkest areas, thereby retaining the subtleties of colors which might otherwise be lost. For more information about UCR and color separations in general, please consult the *Professional Draw* main manual, Chapter 12, page 112.

Since printing presses can only print one color at a time, color separations are required. Knowledge about what settings to use when creating a color separation increases with experience. If you need more information on color separations, such as what densities or angles to use when creating halftone screens, we suggest you talk to your printer or ImageSetter manufacturer.

## Autotiling Output

The *PostScript Output* and *Dot Matrix Output* requesters have been given an updated look and a new feature, *Autotiling*. *Autotiling* enables *Professional Draw* to output pages that are larger than the paper size used in the printer. For example, if the folio page is 15 x 20 inches and the paper size is 8 x 11 inches, only a portion of the folio's image would normally be printed onto the paper. With *Autotiling*, *Professional Draw* will divide the folio page into paper size portions and output as many sheets as necessary to print the entire folio page. These pages can then be assembled to full size.

The *Autotiling* gadget is located in the *Print to PostScript* and *Print to Dot Matrix* requesters.

### To turn autotiling on:

- ➡ Click on the *Autotiling* gadget in the appropriate requester. (*Autotiling* is on when the box beside it is colored in.) Ensure the output page size corresponds to the size of your printer's paper.
- ➡ Click on *OK*. The folio page will now be printed.

## Dot Matrix Offset

The Print to Dot Matrix output requester now has an X/Y page offset option that allows you to offset your document pages from the upper left hand corner of the printed page. Owners of Hewlett Packard LaserJet and DeskJet printers should find this a convenient way to get around those printers' offset problem.

Print to Dot Matrix

From Page 1 To Page 1 # Copies: 1 ☒ Black & White

☐ Current Page ☒ Folio ☐ Grey Scale

Output Scale: X 1.88 Y 1.88 ☐ Color ☐ Correction

Offset X 8.8888 Y 8.8888 Dither: ☐ Ordered

☒ Eject Page ☐ Landscape ☒ Bitmaps ☐ Half tone

Driver: HP\_LaserJet ☐ Floyd-Steinberg

Density: ☐ ☐ ☐ ☐ ☐ ☒ DPI: (300, 300)

☐ Autotile Output Page Size X 8.8888 Y 18.888

OK Cancel

To use the X/Y offset feature, simply enter the X and Y distances from the upper left hand corner of the printed page, in the X and Y gadgets. For example, entering 0.5" for the X offset and 1.2" for the Y offset would cause the top left corner of your document page to begin printing 1/2 inch from the left edge, and 1.2 inches from the top edge of the page. Negative values can also be used to move the image in the other direction.

## HPGL Scaling and Offset

Custom offsets and scaling have been added to the HPGL Output Requester. To set a custom offset and scale, simply disable the Automatic Scale option, and enter your X and Y scale and offsets. These work the same way the Scale and Offset options in the Dot Matrix Output Requester do. (See above and page 106 of the main manual.)



## Undo

Version 3.0 of *Professional Draw* includes an Undo feature that allows you to undo your last action. If you should make a mistake, or if an operation results in an undesirable side effect, simply press *ESC* on your keyboard and the last action you performed will be deleted. Undo works on only the most recent operation, and you can Undo an Undo, effectively replacing the action preceding the Undo.

The following operations may be undone:

- Creating an object
- Deletion of an object
- Repositioning of an object
- Reordering of objects (to front/to back)
- Rotating an object
- Scaling an object
- Aligning groups of objects
- Centering groups of objects
- Cloning an object
- Importing an object or text
- Deleting a page

## Redo

Some operations in *Professional Draw* are remembered and can be repeated by pressing *Shift-Esc*. Pressing this key combination will repeat only the most recent operation. For example, if you draw a rectangle on the page, and then press *Shift-Esc*, a rectangle identical to the one you just drew, will be placed on the page, slightly offset from the original. You can press *Shift-Esc* repeatedly to redo the last operation several times.



# 4. Using Professional Draw 3.0

## Text on a Curve

When using the *Align Text with Curve* item in the *Special* menu, you must select both the text and the curve you wish to wrap it around.

- Select the *Text* tool and type your text.
- Using the *Pen* tool, draw the curve on which you wish to place the text.
- Select both the text and the curve by clicking on the text then holding down the *Shift* key and clicking on the curve.
- Select *Special/Align Text with Curve*.
- Choose the appropriate options in the requester that pops up.

You may wrap text around curves or objects of any shape. For more information on using *Text to Curve* see page 77 of the *Professional Draw 2.0* manual.

## Blend

When you create a blend, two simple objects must be selected. A simple object is an object that contains only one continuous line. Compound images created with *Object/Make Compound Object* cannot be blended.

Many of the effects you created using color blended objects in version 2.0, can be more easily created with *Attributes/Fill/Gradient*. See the above section on *Gradient Fills*.

For more information on using blends see page 78, and the tutorial on page 25 of the *Professional Draw 2.0* manual.

## Add Control Point Tool

If you are having difficulty modifying a curve you can use the *Add Control Point* tool to add extra control points to aid your modifications.

### To add control points:

- Select the *Add Control Point* tool.  
The cursor changes to crosshairs.
- Place the crosshairs on the outline of the object you wish to modify and click the left mouse button.  
A new control point will appear at this position.

- ➡ While holding down the left mouse button, drag the control point to its new position.
- ➡ Release the mouse button.  
The new control point will be moved to the point where you released the mouse button.

The curve of the line will change as you drag it.

During this action, You can use the *Alt* key to constrain the movement to 45 degree angles.

## Cut Control Point Tool

The *Cut Control Point* tool is used to cut lines and objects or to remove a control point.

### To cut an object:

- ➡ Select the *Cut Control Point* tool.
- ➡ Place the cursor on top of the line you wish to cut, and click the left mouse button.  
You can now separate the two pieces of the object by dragging the selected piece away.

### To remove a control point from an object:

- ➡ Select the object.
- ➡ Click on the *Cut Control Point* tool.
- ➡ Place the cursor on the control point you wish to remove and hold down the *Alt* key while clicking the left mouse button.

**NOTE:** When you remove a control point, the line it was on is redrawn between the two points on either end of the line as if the removed control point never existed. This may result in a change in the curve segment's shape.

## Joining Line Segments

### To make two line segments into one line segment:

- ➡ Select the two segments to be joined by clicking on one segment then pressing and holding the *Shift* key while selecting the other segment, or by using the *Marquee* tool.
- ➡ Select the *Pen* tool.
- ➡ Press and hold the *Ctrl* key while drawing a line from the end control point of the first segment to the end control point of the second segment.

The two line segments and the connecting line are now one line segment and be can be moved or altered as such.

**HINT:** If you want to join the two line segments without the connection line, use the *Alt* and *Cut Control Point* tool combination described above to remove one of the connection points after joining the segments.

## The Help Function

You can quickly access information about any tool by selecting the tool and pressing the *Help* key.

## HPGL Plotter Support

The HPGL driver used by *Professional Draw* only supports one pen or knife. Multiple colored pens, plot fills and color fills are not supported when you output to a plotter.

If *Autoscale* is selected some plotters may distort the aspect ratio of the objects in your drawing. Check your plotter's manual, many plotters will allow you to adjust the plot aspect ratio to obtain a more accurate plot. In most cases this is accomplished by adjusting the P1 and P2 points from the plotter's control panel.

## Saving Clips

*Professional Draw* clips are saved in files. A file can contain multiple clips. This is useful for grouping related clips in one file.

### Saving Clips from Professional Draw

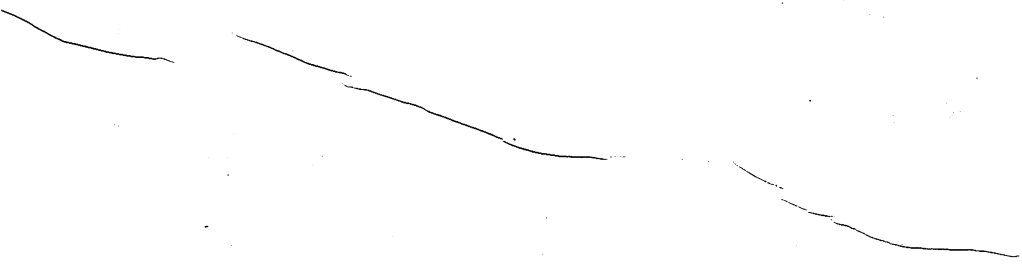
When you select the *Clip/Save* menu command, a requester will pop up, presenting you with a list of clips currently in memory.

➡ Highlight the clips you wish to save, and click on *OK*, then enter the file name.

### Saving Clips from Trace

When you save a clip from *Trace*, you cannot add it directly to an existing Clip file. If you specify an existing file in the *Output File* text gadget, the clip will overwrite any clip already in the file. So if you trace a bitmap of a golf tee, you can call it "Golf Tee" in the *Clip Name* text gadget, but you will have to create a temporary clip file to place it in. You can then modify the tee in *Professional Draw* and place it in the Golf file by loading the Golf file first, then saving.

See Appendix A, page 135, in the *Professional Draw 2.0* manual for more information on *Trace*.



# 5. Function and Tool Genies

*Professional Draw*'s new Function and Tool Genie facilities are useful for automating common operations, creating special effects or making major changes to an existing document. In fact, it is now possible to do virtually anything in *Professional Draw* with just a few clicks of the mouse! A number of sample Function Genies have been included in your *Professional Draw* package.

*Professional Draw* Function Genies are implemented as *ARexx* scripts so they require *ARexx*. Consequently, you must have *ARexx* running and the *rexx:* directory assigned. If the *rexx:* directory is not assigned in your system when you load *Professional Draw*, you will be presented with a requester telling you to "Insert volume *rexx:* in any drive". If you have not run the *RexxMast* program to make *ARexx* available, you will get an error message saying "Genies require *ARexx* to operate" when you start *Professional Draw*. If you get this message, quit *Professional Draw*, start *RexxMast* and restart *Professional Draw*.

Since Function Genies are actually *ARexx* scripts, the Genie facility allows you to run and even edit *ARexx* scripts from within the application itself. For more details of both the implementation and the available *ARexx* commands, please see Chapter 6, "*ARexx Support and Commands*", of this Addendum.

## Executing a Function Genie



When you select the *Function Genie* tool a list box will appear with the list of available Function Genies and a group of several buttons. The Function Genies displayed in the list box are not in memory, but are on disk (in *rexx:*). To execute a Function Genie, simply double-click on it (or select it from the list and click on the *Execute* button).

## Using the *About* Button

Information about the use of each Function Genie is available. After you select the Function Genies Tool and the list box shows the available Function Genies, simply select the desired Genie with the mouse and click on the *About* button. A window will open and the information for that Function Genie will be displayed.

## Defining and Modifying Genies

New Function Genies may be created by clicking on the *Define* button. This will bring up the *Genie Editor* so that you can enter the *ARexx* script for your Function Genie. You may also edit existing scripts by selecting one from the list box and clicking on the *Modify* button. If you are using an editor other than the Genie Editor, do not use any style changes (i.e., bold, italic) as *ARexx* will consider them to be errors.

When you have finished editing, save the file and select *Return* from the *Project* menu to return to the main **Professional Draw** window. The Genie will automatically be saved to the *rexx:* directory. You need not append a suffix to the name as **Professional Draw** will automatically append a *.pdrx* extension (which will not appear in the Genie list).

When creating a new Function Genie, you must begin with a comment line. *ARexx* requires this. A comment in *ARexx* begins with “/\*” and ends with “\*/”. Here are some examples:

```
/* This is an ARexx comment. */  
/*
```

The comment need not be on a single line. It just needs to start and end with the right characters.

```
*/
```

It's a good idea to use the comment line for describing the program's function since the information contained in this initial comment is displayed when you click on the *About* button.

## Function Genie List Management

The *Install\_PDraw* program will place the Function Genies into the *rexx:* directory. When you select the *Function Genie Tool*, the list box will display that list of the available Function Genies. However, if you have Genies in a directory other than *rexx:*, you may still access them by clicking on the *Import* button which will bring up a file requester. Importing a Function Genie in this way will make it available by copying it to the *rexx:* directory.

The *Delete* button allows you to delete a Function Genie. However, you should be aware that *it is the Function Genie file on disk that is actually deleted* and, once deleted, the file is not recoverable.



## Function Genies Keys

You may assign a function key shortcut to any Function Genie by selecting the desired Function Genie and clicking on the *Keys* button. **Professional Draw** will then prompt you to press a Shortcut key. This must be either the *Alt* key in conjunction with one of the function keys or *Alt-Shift* in conjunction with a function key. The keyboard shortcut associated with a Function Genie (if any) is displayed to the right of the Genie's name. To erase a shortcut assignment, simply select the Function Genie, click on *Keys*, and hit the *Esc* key.

## Unexpected Returns

If *ARexx* encounters any problems when a Function Genie is executed, a *Results from ARexx command* screen will pop up containing the *ARexx* messages and/or errors generated by the execution of the Function Genie. You are most likely to see this screen when developing new Genies.

If there are more messages and/or errors than will fit on the screen, the word "[MORE]" will appear at the bottom right of the screen. You may use the cursor keys or the left mouse button to scroll through the list. When you are finished reading your results, you can close the screen by pressing the 'q' key, the *Return* key or by pressing the right mouse button.

## The Function Genies

The remainder of this section is devoted to a description of the purpose and operation of the Genies that are included in your **Professional Draw** package. Note that many of these Genies will put prompts and status reports into the **Professional Draw** title bar and consequently you must be attentive to changes in the title bar.

Some Genies require the use of *gdarexxsupport.library*. This file is placed into your *libs:* directory automatically when you run *Install\_PDraw*. If the file is removed from your *libs:* directory, you will get an error message from any Genie that requires the use of this library. Copying the *gdarexxsupport.library* file to your *libs:* directory will allow you to continue.

### **AddToLabelDataBase**

This Genie will allow you to add label sizes to **Professional Draw**'s Avery label database.

Enabling this Genie will bring up a series of requesters allowing you to specify the label's part number, size (will default to the size of the currently selected object), type (laser or dot matrix), number of columns (laser), number of rows, margins, horizontal pitch (distance between the left edges of side by side labels on the sheet), and vertical pitch (distance between the tops of labels in one row and in the next).

## **AdjustColors**

This Genie will adjust the colors of selected objects. You may adjust the RGB or the CMYK values of the fill and outlines of the selected objects. When prompted, input the percentage change in the value for each color you want to adjust. To adjust colors, select the objects to be adjusted, and enable the Genie. A requester appears allowing you to adjust the Hue, Saturation, and Value for the colors. Adjust the percentages as required, 100 % is full intensity, and 0% is white.

## **ADProHotLink**

This Genie will run *Art Department Professional* (published by ASDG Incorporated) and pass a bitmap graphic to it from your **Professional Draw** document. First, you are prompted on the title bar to select the bitmap object. Click on the desired object. If it is not a bitmap, you will get an error and the Genie will terminate.

When a valid object is selected, you will be presented with a list box that contains the *ADPro* screen modes: HIRES, V\_OVERSCAN, INTERLACE, PAL and H\_OVERSCAN. Select the desired mode(s) and click on *OK*. Next, you must select the Render Mode for *ADPro*. Click on your choice and click on *OK*. The next requester prompts for the image dimensions. The values shown in the *Width* and *Height* string gadgets represent the size of your bitmap in pixels.

The Genie will then try to run *ADPro*. If the full path to *ADPro* is *ADPro:ADPro*, it will run automatically; otherwise, a file requester will appear so that you can tell the Genie where to find *ADPro*.

You must save your changes to the bitmap file and quit *ADPro* to continue using **Professional Draw**. The Genie will sleep until you quit *ADPro*; at which time, your graphic will be returned to its box in the **Professional Draw** document while “Reimporting bitmap...” appears in the title bar.

## **AutoSave**

This Genie can spare you a lot of grief. It will prompt you to save your work at regular intervals and, optionally, create a backup file for you. The backup file is the *original* copy of the file for this session. For example, if you run **Professional Draw** and load a document file called *MyDoc*, a backup file of that version of the document is made (*MyDoc.bak*) and each time you save, the file *MyDoc* is updated. For the duration of that session, *MyDoc.bak* will remain the same. The next time you run **Professional Draw** and work with *MyDoc*, a new *MyDoc.bak* file will be created.

When the Genie is executed a requester will present you with two prompts: one for the number of minutes to wait before prompting you to save and one requesting whether the Genie should make a backup file. Enter appropriate responses and click on *OK*.

When the specified interval has elapsed, a dialog box will appear on the screen, asking if you want to save now. Clicking *Yes* will save your document and, if you have directed the Genie to do so, a backup file will be created. Clicking on *No* will bypass the save operation but leave the Genie running. Clicking on *Cancel* will terminate the *AutoSave* Genie. A backup file will only be made the first time you save the document. This way you can always revert to the version of the artwork that you started the session with.

If you wish, you can use the *SavePrefs* Genie (see below) to make the *AutoSave* Genie run automatically every time you run ***Professional Draw***.

### ***AveryLabels***

This Genie will come in very handy if you often find yourself generating labels for mailing lists, file folders, audio cassettes, video cassettes, floppy disks - most anything! This Genie creates pages to exactly match labels produced by Avery, both those intended for dot-matrix printers (tractor feed, fanfold) and those intended for laser printers (full page sheets).

When you invoke this Genie, a list box appears containing the names of the various label types: Dot Matrix Labels, European Laser Labels, French Dot Matrix Labels and Laser Labels. Select one of these and click on *OK*. "Reading label database.." will appear in the title bar. Once the database has been read, it will be displayed to you in the *Select Label* list box. Each entry consists of the Avery catalog number followed by a short description of the type of label. Click on the type of label you're using. If a box or group is currently selected, you will be asked if you want it to be scaled to fit the dimensions of the label. Click on *Yes* or *No*, and the matching page will be created.

If you've chosen Dot Matrix Labels, each page corresponds to only one label and you will be asked, "How many Dot Matrix Labels?" The number you enter is the number of 'pages' (i.e. labels) that will be created. This is convenient when you have a fixed, known number of labels to generate.

### ***ClickRadFill***

This Genie allows you to set the center of a radial fill by clicking. Select the object you want to fill, and enable the Genie. Click on the page, where you want the center of the fill to be located.

### ***Command***

This Genie allows you to issue *ARexx* commands directly. This is only recommended for advanced users. If the command takes no parameters, the empty parentheses may be left off and the Genie will append them automatically.

### ***ConvertToBW***

This Genie will convert selected objects to Black and White. Select the objects you wish to convert and enable the Genie. You will be presented with a threshold value. All color intensities in the selected objects that are greater than this value will be converted to Black. All intensities lower than the threshold value will be converted to white. You may edit the threshold value if you wish. Click on *OK* and the objects will be converted.

### ***ConvertToGray***

This Genie will convert the currently selected objects to shades of gray. Simply select the objects you wish to convert, and enable the Genie. The Genie automatically takes the color intensity values and converts the colors to gray shades.

### ***CopyObjectAttr***

This Genie will copy the attributes of one object to other selected objects. Select the objects you wish to copy attributes to, and enable the Genie. You will be prompted to click on the object you want to copy attributes from. After you click on the object, a requester will appear allowing you to specify what attributes you wish to copy. Select the attributes and click on *OK*.

### ***CopyObjToPages***

This Genie will replicate selected objects across a range of pages. Select the objects you wish to copy attributes to, and enable the Genie. An *Enter options* requester will appear, prompting you for the first and last page number defining the destination range. You also have the option of copying the selected objects to odd, even or all pages.

### ***CopyPages***

This Genie will create new pages from a given range of pages. A requester appears requesting input for **from**, **to** and **number of copies**. As an example, if you have a one-page document and you input "1" for from, "1" for to and "3" for number of copies, the result will be a four-page document in which the last three pages are duplicates of the first. Any existing pages following the source range will be pushed forward to higher page numbers to make room for the insertion of the copied pages.

### ***DeleteRange***

This Genie deletes a range of pages. Simply enter the page numbers corresponding to the first and last pages of the range to be deleted. As you would expect, any existing pages following the deleted range are pulled back to lower page numbers, replacing the deleted range.

### ***DistributeObject***

This Genie will distribute objects over a user- selected distance (both horizontally and vertically). This is particularly convenient when you have created an object and cloned a number of new objects from it. Select the group and invoke the Genie. You will be selecting the manner of distribution and the amount of space.

The *Horizontal Distribution* requester offers five choices: Objects, Centers, Left, None and Right.

Objects will divide the total space into equal parts. The space between each of the objects will be equal to one of those parts. Centers will position the object centers evenly over the specified space.

Left and Right will position the specified object edges evenly over the specified space.

None will result in no horizontal displacement of the objects in the group.

The *Vertical Distribution* requester is the same as the horizontal one except that it works in the other dimension. When you have selected from both requesters, a new requester prompts you for *From* and *To* values. These values define the space within which the group will be distributed. Enter appropriate values to define the desired area on the page.

### ***ObjectToLayer***

This Genie will move selected objects to a specific layer on the page.

After the objects you want to move are selected, enable the Genie. A request appears showing what layer you are moving the objects from, and the highest possible layer you can move them to. Enter the layer number you want to move the objects to in the *To* field and click on *OK*. The objects will be moved.

### ***PointsAlign***

This Genie will align selected points horizontally, to the top, center or bottom, and vertically to the left, right or center. Simply select the object on which you want to align some of the points, and enable the Genie. Next, you select the points you wish to align (clicking anywhere on the page where there is not a point to end the selection), and specify where you want the points aligned to.

### ***PointsMove***

This Genie will move selected points to a specified position, Simply select the object or objects on which you want to move some of the points, and enable the Genie. Select the points you wish to move (clicking anywhere on the page where there is not a point to end the selection). A requester will appear asking for the X and Y co-ordinates that you want the points moved to.

### ***PointsOffset***

This genie will offset points on a bezier object by a user specified amount.

Select the bezier object on which you want to offset some of the points, and enable the Genie. Next, you select the points you wish to offset (clicking anywhere on the page where there is not a point to end the selection). A requester appears asking you for the X and Y offset. Enter these and click on *OK*. The points will be offset from the rest of the object by the specified amount.

### ***SavePrefs***

This Genie will bring up a list box containing all the settings available in ***Professional Draw***. Click on all the ones you would like to preserve for future sessions. The Genie will save your current settings to a file (*s:pdraw.config*). Each time you run ***Professional Draw***, the configuration file will be loaded and your specified settings will be restored. In addition, this Genie will ask you if you want to run the AutoSave Genie each time that ***Professional Draw*** is run.

### ***ScaleToSize***

This Genie will scale an object to a new size. A *Select Scaling Options* requester will prompt you for a new width and height.

### ***SpaceObjects***

This Genie will space objects both horizontally and vertically, based on user input. This is convenient when you have created an object and cloned a number of new objects from it. Select the objects and invoke the Genie. A requester appears allowing you to select the manner of distribution and the size of the space between objects.

### ***SpacePoints***

This Genie will space points on a bezier object, both horizontally and vertically, based on user input. Select the points you wish to space, and invoke the Genie. A requester appears allowing you to select the manner of distribution and the size of the space between points.

### ***StepAndRepeat***

This Genie will duplicate, move and rotate an object according to your instructions. Select the object you want to use and enable the Genie. The *Step & Repeat* requester then appears, prompting you for: Count, Horizontal Offset, Vertical Offset and Angle. Count is the number of copies that will be created. The Horizontal and Vertical Offset values are added to the left and top values of the previous object in the chain to set the position of the next object. Similarly, for each new object, the specified Angle value is added to the Rotation Angle of the previous object in the chain.

### ***StyleTagAdd***

This Genie will add a set of object attributes to a Style database. The attributes may come from the current menu settings, or the currently selected object

Once the attributes are selected, (by selecting them in the menus, or selecting an object that contains the attributes) enable the Genie. A requester will appear allowing you to specify what attributes you wish to include in the style. Choose the attributes and click on *OK*. You will then be asked to name the new style.

### ***StyleTagApply***

This Genie will apply a selected style to selected objects. Select the objects you wish to apply the style to, and enable the Genie. A requester appears containing a list of the existing styles. Select the style you wish to apply, and click on *OK*. The selected objects will be filled according to the selected style.

### ***StyleTagDelete***

This Genie will delete styles. When this Genie is selected, a list of all defined styles will be displayed. Select the Genie or Genies you wish to delete, and click on *OK*. The selected Genie will be deleted permanently.

### ***TileSelection***

This Genie will replicate a group until it fills the page. This is a quick and easy way to create a tiled backdrop.

Select the objects you want to tile, and enable the Genie. An Enter Coordinates requester will appear containing string gadgets for: Start X, Start Y, Number of Columns, Number of Rows, Horizontal Spacing, and Vertical Spacing. The gadgets will automatically display values to fill the current page. However, you are free to alter them if you wish. Click on *OK* when all coordinates are acceptable. The group will be tiled on your page. Note that the group is moved to the top left corner of the page before it is tiled.

### ***UnitsConverter***

This Genie will convert values from one typesetting unit to another. The equivalencies are based on 1 inch = 72 points, 6 picas, 2.54 cm, 66.9566 ciceros, or 14 agates. The *Conversion yields...* requester appears. Enter a value for any unit and click on *OK*. The requester will reappear containing the converted values. You may then enter another value for conversion and continue in this fashion until you click on *Cancel* or click on *OK* without making any changes.

## Special Effect Genies

There are also 4 Genies that will give you an idea of some of the fun things you can do using ARexx functions to create your own Genies.

### ***Flower***

This Genie draws flower-like objects. You specify the number of petals (sides on the initial object), and how sharp or rounded the petals will be. The Genie then creates 10 scaled copies of the object, and places them on top of each other to create a flower-like object.

### ***Mandela***

This Genie creates a number of polygons placed on top of each other, each slightly rotated from the previous one. You specify whether you want to use a custom shape (the currently selected object) or a circle, and the number of objects, and their radius.

### ***PlotFunction***

Genie will plot a mathematical function in terms of X and Y. After you enable the genie you will be asked to specify the start point, the stop point, the number of points, and the mathematical function in a requester. Specify these, and click on *OK*. The function will be plotted.

### ***PlotPolar***

This Genie will plot a polar mathematical function in terms of angle 'a'. After you enable the genie you will be asked to specify the start angle, the end angle, the number of points, and the polar mathematical function in a requester. Do this and click on *OK*, the function will be plotted.



## Tool Genies

*Professional Draw 3.0* also has 4 tool genies that give you more flexibility in drawing and arranging objects. These Genies are enabled by double clicking on the appropriate tool.

### Executing a Tool Genie

Tool Genies exist for the *Marquee*, *Pen*, *Ellipse*, and *Rectangle* tools. To execute a Tool Genie, simply double click on the appropriate tool. This will bring up the necessary requesters to execute the Genie

#### ***Marquee Tool***

This Genie will allow you to select objects by a wide variety of criteria such as size, color, type, layer et cetera. Double clicking on the *Marquee tool* will bring up a requester listing the criteria you can base your selection on. Select the appropriate criteria (you can make multiple selections). A series of requesters will appear asking you for the specifics of your criteria selection. For example, if you wish to select objects by color, a second dialog box will appear listing colors, so you can choose what colors of objects you wish to select.

#### ***Rectangle Tool***

This Genie allows you to create rounded corners on the selected rectangles. Double clicking on the *Rectangle tool* will bring up a requester allowing you to specify the radius of the corners of the rectangle. Entering a negative radius or a radius that is more than half the length of the rectangle's side, will create interesting effects.

#### ***Ellipse Tool***

This Genie allows you to specify the size of an ellipse numerically. Double clicking on the *Ellipse tool* will bring up a requester allowing you to specify the height and width of the ellipse to be drawn.

#### ***Pen Tool***

This Genie allows you to create multiple sided polygons. Double click on the *Pen tool*. A requester will appear allowing you to specify the number of sides on the polygon, and the polygon's radius. Enter these values and click on *OK*. The polygon will be drawn.

### Creating and Altering Tool Genies

Because Tool Genies are accessed from the tool palette and not through the Function Genie requester, they cannot be edited in the same way that Function Genies are. You can still edit them to suit your needs with any standard text editor, such as Workbench's Ed program. Tool Genie files are prefixed with `_PD_TOOLS_`. You can

also add Genies to the *Freehand* tool, *Cut Control Point* tool, and the *Add Control Point* tool. These Genies should be created with the filenames:  
\_PD\_TOOLS\_FREE.pdrx, \_PD\_TOOLS\_CUTP.pdrx, \_PD\_TOOLS\_ADDP.pdrx  
respectively.

**NOTE:** Any Genie name that begins with "\_" will not appear in the Genie list when you click on the *Function Genie* tool.

## 6. **ARexx Commands and Support**

*Professional Draw*'s *ARexx* interface recognizes over 300 commands. Virtually every aspect of *Professional Draw* operations may now be driven by an *ARexx* script. If you have not been using *ARexx* previously, we strongly recommend that you obtain and install it on your system since many programs, not just *Professional Draw*, can take advantage of its presence to offer you additional capabilities. Depending on your circumstances, you may wish to use *Professional Draw* in conjunction with other *ARexx*-aware applications, such as an image-processing program or a terminal program.

Once you have obtained and installed *ARexx*, it is also a good idea to insert the line  
`RexxMast`

somewhere into your startup-sequence so that *ARexx* will always be available.

If you're new to *ARexx* programming (or even if you're not), we recommend that you obtain a copy of the book *Using ARexx on the Amiga* by Chris Zamara and Nick Sullivan. The book contains both tutorial and reference material and includes a companion disk. *Using ARexx on the Amiga* is published by Abacus.

### **Technical Information**

The *Professional Draw* *ARexx* host is implemented as an *ARexx* *function* host (rather than the more usual *command* host). This is because many of the *ARexx* commands in *Professional Draw* return a result, and results can be handled much more tidily when they appear as a function return than when they must be accessed through the `RESULT` keyword.

One important characteristic of a function host is that it is *global*; every function call in every *ARexx* macro is passed through a multi-stage search procedure that begins with internal functions (those in the macro source itself), continues with the built-in *ARexx* function library (the functions listed in the *ARexx* manual), turns next to function hosts (such as *Professional Draw*) and function libraries, and finally to external programs on disk. Consequently, it is not necessary to address *Professional Draw*'s specific *ARexx* port ("PDRAWAREXX").

All resident *Professional Draw* *ARexx* functions may be called from within *Professional Draw* Function Genies by their listed names. If you are calling them from an external *ARexx* program then you *must* prefix the names with "PDM\_" in order for

the calls to work. Because function calls are global rather than being addressed to a particular host, there is a danger of inadvertent *name collision*. The prefix ensures that your function will be received by **Professional Draw** and not by some other program that has a function with the same name.

## Parameter & Return Information

The format of the commands is:

```
* result = Function(arg1, [arg2...])
```

When an argument is optional, it is enclosed in square brackets in the definition. If optional arguments are omitted, you must still include the commas.

An asterisk at the beginning of the line signifies that the function may involve some user interaction. This interaction may depend on whether optional arguments are included or not. If optional arguments are omitted, they will be set interactively through the usual requesters.

If a function returns more than one item of data, those items are returned in a string with spaces or carriage returns separating each item. Argument and return types are discussed below for each function.

### Notes:

- All **Getxxx** functions return data in the same form as the corresponding **Setxxx** function.
- Numerical parameters are in current user units (eg. inches) unless otherwise noted.
- Functions that do not return a value *must* be preceded by a "call" in *ARexx*; eg., call **AutoUpdate**(0).
- Arguments consisting of text should be enclosed in double quotes; eg., call **ShowStatus**("This is text").

## Page and Object Identifiers

In **Professional Draw 3.0** pages and objects have unique identifiers to allow macro access. Such an identifier may be either a user-specified name or number. If an identifier is optional and absent in the command, the current page (or object(s)) is/are assumed.

**Note:** In the material that follows <CR> represents a Return character or new line. This is specified in *ARexx* by ' 0a' x.

## General Functions

call **SetBatchMode** ([bool])

When *bool* is 1, all requesters will automatically be responded to with *OK* or *Resume*, so that no requesters are displayed on screen unless explicitly called by an ARexx command. When *bool* is 0, all requesters will be displayed normally.

call **AbortPrint** ()

Stops current printing.

call **PDrawToFront** ()

Brings the *Professional Draw* window to the front.

call **PDrawToBack** ()

Sends the *Professional Draw* window to the back.

call **UpdateScreen** ([bool])

Redraws the entire screen when *bool* is 0. Redraws only the changed areas of the screen when *bool* is 1.

call **ZoomScreen** ([pointX], [pointY], [width])

Magnifies the screen centered at *pointx*, *pointy* to a width of *width*.

**AutoUpdate** (bool)

Clears (when *bool* is 0) or sets (when *bool* is 1) auto-update mode. When auto-update mode is set, after each ARexx function, the screen is updated (if there are no pending messages). Most of the time you will want to turn auto-update mode off. When the parameter is (0), the screen is not updated until the Genie is finished or an *UpdateScreen* command is issued.

call **Pause** (secs)

Pauses for the number of seconds indicated by *secs*.

totalpages = **NumPages** ()

Returns the number of pages in the document.

numobj = **NumPageObjs** ([page])

Returns the number of objects on page *page*. If *page* is not specified, returns the number of objects on the current page.

totalobj = **NumDocObjs** ()

Returns the number of objects in the document.

numactobj = **NumSelObjs** ()

Returns the number of objects currently selected.

`numobj = NumGroupObjs([obj])`

Returns the number of objects in the group that contains object *obj*. If *obj* is not specified, returns the number of objects in the currently selected group.

`numobj = NumCompObjs([obj])`

Returns the number of objects in the compound object that contains object *obj*. If *obj* is not specified, returns the number of objects in the currently selected compound object.

`change = DocChanged()`

Returns a 1 if the document has changed since the last save, 0 if not.

`call ShowStatus(text)`

Displays *text* on the **Professional Draw** title bar. This function can be used to prompt the user for mouse clicks, etc.

`call ClearStatus()`

Restores the title bar to the normal **Professional Draw** title.

`call SaveText(file, text)`

Creates a *file* and saves *text* into it.

`call SaveMoreText(file, text)`

Appends the given *text* to an existing *file*.

`call EditText(file)`

Brings up the Genie Editor to edit *file*.

## User Interactive Functions

`* userInput = GetUserText(size, prompts)`

Gets a single line of input, with a maximum length of *size*, from the user using *prompts* as a directive.

`* manylines = GetForm(title, size, prompt)`

This function will return a formatted string containing multiple lines of user input. The *title* string will appear in the requester's drag bar. The length of the requester's string gadgets are specified by *size*. The formatted string is defined by *prompts*. If you wish, you may also use *prompts* to fill the string gadgets with a default response; eg.,  
Prompt1:default1<CR>Prompt2:default2<CR>...

`* choiceint = Inform(numbuttons, text, [button1txt, _  
_[button2txt, [button3txt]]])`

This function will prompt the user with *text* to select 1 of up to 3 choices. The *numbuttons* parameter is one of 1/2/3. If no button text is specified, a value of 1 will use *Resume* for the button text and a value of 2 will use *Yes* and *No*. Note that the first

button is positioned in the lower *right* of the dialog box, the second to the *left* of the first, and the third to the left of the second. The function returns -1 if there is an error, 0 if the first (rightmost) button is clicked, 1 if the second, and 2 if the third.

```
* filespec = GetFileName(title, path, [filename])
```

This function brings up a file requester and returns the file specification. If the user cancels the requester, an empty string is returned.

```
* chosen = SelectFromList(title, width, height, mode (0 through 3), list)
```

A method of offering the user a list of multiple choices and receiving the user's response. *Width* is the width of the list in characters, *height* is the height in lines. When *mode* is 0, the list is sorted and a single choice is returned. When *mode* is 1, the list is sorted and multiple selections are returned. When *mode* is 2, the list is not sorted and a single choice is returned. When *mode* is 3, the list is not sorted and multiple selections are returned. Both *list* and *chosen* are listed in the form of: Item1<CR>Item2<CR>... Items preceded by an underscore, "\_", will be preselected. The "\_" will not be displayed in the list.

```
* whichobj = ClickOnObj([title])
```

Displays *title* and waits for the user to click the left mouse button on an object and returns the object ID of the selected object. This allows Genies to operate on selected objects without having to know the object name or ID. The function will return 0 if the user clicks on a location that does not contain an object.

```
* whichobjpoint = ClickOnPoint([title])
```

Displays *title* and waits for the user to click the left mouse button on a control point of an object, and returns the object ID and the point number of the selected point. This allows Genies to operate on individual control points. The function will return "0 0" if the user clicks on a location that does not contain a control point.

```
* mouseXY = ClickRectangle([prompt], width, height)
```

Displays *prompt* and a rectangular pointer that is *width* wide and *height* high, and waits for the user to click the rectangle at a point. The function then returns the position of the mouse where the user clicked.

```
* mouseXY = ClickEllipse([prompt], [width], [height])
```

Displays *prompt* and an elliptical pointer that is *width* wide and *height* high, and waits for the user to click the ellipse on a point. The function then returns the position of the mouse where the user clicked.

```
* whicharea = ClickArea([prompt])
```

Displays *prompt* and waits for the user to drag a rectangle. The function returns the left, top (X,Y position of top left corner), width, and height of the area.

\* `mouseXY = GetClickPosition([prompt])`

Displays *prompt* and waits for the user to click the left mouse button and returns the X,Y position of the location on the page where the user clicked.

## Project Menu Commands

\* `success = LoadDocument([file])`

Loads the specified file into *Professional Draw*.

\* `success = SaveDocument([file])`

Performs a *Save* operation using the current name. If *file* is supplied, a *Save As* operation is performed, renaming the document to *file*.

`datewhen = CreationDate()`

Returns the date on which the file was created.

`lastsave = SavedDate()`

Returns the date of the last *Save* operation for this document.

`datenow = CurrentDate()`

Returns the current date.

`whodoc = GetAuthor()`

Returns the author's name as entered in the Information box.

`comment = GetComment()`

Returns the comment as entered in the Information box.

`namedoc = GetDocName()`

Returns the name of the document as entered in the Information box.

`revisenum = GetRevision()`

Returns the revision number as entered in the Information box.

\* `call SetDocInfo()`

Brings up the Project Information requester.

`call SetAuthor(name)`

Replaces the author's name in the Information box with *name*.

`call SetComment(comment)`

Replaces the comment in the Information box with *comment*.

`call SetDocName(name)`

Replaces the document name in the Information box with *name*.



call **SetRevision**(revisenum)

Replaces the revision number in the Information box with *revisenum*.

\* **bmid** = **Importbm**([bitmapname], [Xpos], [Ypos])

Places the bitmap, *bitmapname*, at position *Xpos*, *Ypos* and returns the new object ID. If the *Xpos* and *Ypos* are not specified, the user must place the bitmap with the mouse.

\* **epsfid** = **ImportEPSF**([epsfname], [Xpos], [Ypos])

Places the EPSF file, *epsfname*, at position *Xpos*, *Ypos* and returns the new object ID. If the *Xpos* and *Ypos* are not specified, the user must place the EPSF file with the mouse.

call **New**()

Behaves exactly as does selecting the menu item *Project/New*.

call **PDQuit**()

Exits *Professional Draw* (same as selecting the *Project/Quit* menu item).

## Page Operations

**cpage** = **CurrentPage**()

Returns the page number of the current page.

**newpage** = **GotoPage**(page)

Sets the current page to *page*.

**fpage** = **DocFirstPage**()

This function returns 0 if the document contains no pages; otherwise, returns 1.

**npage** = **DocNextPage**([page])

Returns the page number of the document page following the indicated one or, if *page* is not supplied, the page after the current page.

**lpage** = **DocLastPage**()

Returns the page number of the last page of the document.

**page** = **PageUp**()

Displays the next page of the document (like using the Page Up gadget).

**page** = **PageDown**()

Displays the previous page of the document (like using the Page Down gadget).

\* **success** = **LoadPage**([file])

Loads the page specified by *file*.

\* **success** = **SavePage**(page, [file])

Saves the specified *page* using *file* as the filename.

`page = CreatePage(numpages, finishpage, [name] )`

Creates *numpages* number of default pages which are inserted before page *finishpage*. Returns the page number of the first page created. The *name* parameter gives a name to page *fromnumber*.

`numgone = DeletePage([startpage], [numdel])`

Deletes *numdel* pages starting from *startpage*. Returns the number of pages deleted. With no parameters, deletes current page only.

`numcopies = CopyPage(pgnm, pgnm2, [num])`

Insert *num* copies of *pgnm* before *pgnm2*. Returns number of pages created.

`newpage = MovePage(pgnm, pgnm2)`

Moves *pgnm* to the position before *pgnm2* and returns the resulting page number.

`pmargs = GetPageMargins([pgnm])`

Returns the margins of the specified page or the current page as “left top right bottom”.

`call SetPageMargins(pgnm, left, top, right, bottom)`

Sets the margins for the specified page.

`psize = GetPageSize([pgnm])`

Returns the size of the specified page or the current page.

`call SetPageSize(pgnm, Xsize, Ysize)`

Sets the dimensions of the indicated page.

`pname = Getpagename([pgnum])`

Returns the name of the specified page or the current page.

`call SetPageName(pgnum, name)`

Assigns a name to a specific page.

## Object Operations

`center = GetCenter([object])`

Returns the co-ordinates of the center of selected object or objects.

`objID = SelectObj(ID1, [ID2])`

Makes the object *ID1*, or all objects whose IDs fall between *ID1* and *ID2*, the selected objects, and returns the object ID of the first selected object.

`objID = SelectAnother(ID1, [ID2])`

Adds the object *ID1*, or all objects whose IDs fall between *ID1* and *ID2*, to the selected objects, and returns the object ID of the first selected object.

`ObjID = SelectAll()`

Selects all objects on the current page and returns the selected object's ID.

`call CompoundObj([obj])`

Makes selected objects (and object *obj* if specified) into a compound object.

`call GroupObj([obj])`

Makes selected objects (and object *obj* if specified) into a grouped object.

`call AlignObj ([obj], side)`

Aligns selected objects (and object *obj* if specified) by *side*. *Side* can be 0 (top), 1 (bottom), 2 (left), or 3 (right).

`call CenterObj ([obj], dir)`

Aligns selected objects (and object *obj* if specified) by *dir*. *Dir* can be 0 (horizontal), 1 (vertical), or 2 (both).

`* NewObj = CloneObj ([obj], [centerX, centerY, shiftX, shiftY, ScaleX, ScaleY, angle])`

Copies object *obj*, and place the center of the copy at *centerX*, *centerY*, shifting the new object *shiftX* and *shiftY*, and scaling the object by *scaleX* and *scaleY*. The new object will be placed on an angle of *angle* degrees. If *obj* is not specified the currently selected objects will be copied. Returns the new object ID.

`* call = ScaleObj ([obj], [scaleX, ScaleY, Xpoint, Ypoint])`

Scales the selected objects, or just object *obj* by *scaleX*, *scaleY*, about the point *Xpoint*, *Ypoint*.

`* call = RotateObj ([obj], [rotate, [Xpoint, Ypoint]])`

Rotates the selected objects, or object *obj* by *rotate* degrees around point *Xpoint*, *Ypoint*.

`* call = MirrorObj ([obj], [Xpoint1, Ypoint1, Xpoint2, Ypoint2])`

Mirrors the selected objects, or object *obj*, around the line between *Xpoint1*, *Ypoint1* and *Xpoint2*, *Ypoint2*.

`call = MoveObj ([obj], [X], [Y])`

Moves the selected objects, or object *obj*, delta *X*, *Y*.

`* call = BlendObj ([numblend, type])`

Blends two selected objects, creating a total of *numblend* objects. *Type* can be set to: 1 (linear), 2 (sinusoidal), 3 (inverse), or 4 (cubic).

call **DeleteObj** ([obj])

Deletes selected objects (or object *obj* if specified).

call **ObjToFront** ([obj])

Brings selected objects (or object *obj* if specified) to the top of the drawing order.

call **ObjToBack** ([obj])

Sends selected objects (or object *obj* if specified) to the bottom of the drawing order.

call **ObjForward** ([obj])

Brings selected objects (or object *obj* if specified) forward one level in the drawing order.

call **ObjBackward** ([obj])

Sends selected objects (or object *obj* if specified) backward one level in the drawing order.

call **ObjInFront** ([obj], obj1)

Brings selected objects (or object *obj* if specified) in front of *obj1* in the drawing order.

call **ObjBehind** ([obj], obj1)

Sends selected objects (or object *obj* if specified) behind *obj1* in the drawing order.

call **UnselectObj** ([obj])

Deselects selected objects (or object *obj* if specified).

call **UncompoundsObj** ()

Uncompounds selected compound objects.

call **UngroupObj** ()

Ungroups selected grouped objects.

call **SendHome** ()

Sends selected objects to *Professional Page* via the hot-link.

firstobj = **ArtFirstObj** ()

Returns the object ID of the first object on the art board.

nextobj = **ArtNextObj** ([obj])

Returns the object ID of the object after *obj* (or after the currently selected object) on the art board.

lastobj = **ArtLastObj** ()

Returns the object ID of the last object on the art board.

firstobjdoc = **DocFirstObj** ()

Returns the object ID of the first object in the document.

`nextobjdoc = DocNextObj([obj])`

Returns the object ID of the object after *obj* (or the currently selected object) in the document.

`lastobjdoc = DocLastObj()`

Returns the object ID of the last object in the document.

`firstobjpg = PageFirstObj()`

Returns the object ID of the first object on the page.

`nextobjpg = PageNextObj([obj])`

Returns the object ID of the object after *obj* (or the currently selected object) on the page.

`lastobjpg = PageLastObj()`

Returns the object ID of the last object on the page.

`firstobjsel = SelFirstObj()`

Returns the object ID of the first currently selected object.

`nextobjsel = SelNextObj([obj])`

Returns the object ID of the currently selected object after *obj* (or the currently selected object).

`lastobjsel = SelLastObj()`

Returns the object ID of the last currently selected object in the document.

`firstobjgrp = GroupFirstObj([obj])`

Returns the object ID of the first object in the group containing *obj*, or the currently selected group.

`nextobjgrp = GroupNextObj([obj])`

Returns the object ID of the object after *obj* in the group containing *obj*.

`lastobjgrp = GroupLastObj([obj])`

Returns the object ID of the last object in the group containing *obj*, or the currently selected group.

`firstobjcmp = CompFirstObj([obj])`

Returns the object ID of the first object in the compound object containing *obj*, or the currently selected compound object.

`nextobjcmp = compNextObj([obj])`

Returns the object ID of the object after *obj* in the complex object containing *obj*.

`lastobjcmp = CompLastObj([obj])`

Returns the object ID of the last object in the complex object containing *obj*, or the currently selected complex object.

`obj = ObjAtPosn(Xpoint, Ypoint, [page])`

Returns the Object ID of the object that contains point *Xpoint*, *Ypoint*.

## Object Attribute Operations

`call SetObjName(obj, name)`

Sets the name of object *obj* to *name*.

`call SetObjSize(obj, width, height)`

Scales object *obj* to *width*, *height*.

`call SetObjPosn(obj, X, Y)`

Moves object *obj* to co-ordinates *X*, *Y*.

`call SetObjVisSize(obj, width, height)`

Scales object *obj* to *width*, *height*.

`call setObjVisPosn(obj, X, Y)`

Moves object *obj* to co-ordinates *X*, *Y*.

`call SetObjLock([obj], lockstatus)`

Locks or unlocks *obj* (or the currently selected object if *obj* is not specified). When *lockstatus* is 1, the object is locked, when *lockstatus* is 0 the object is unlocked.

`call SetObjPage([obj], page)`

Moves *obj* (or the currently selected object if *obj* is not specified) to page *page*.

`* call SetLineColor([obj], [colname])`

Sets the color for lines. *Colname* must be the name of a color defined in the current color list or an unnamed color specified in one of 3 formats: *r*, *g*, *b*, *y*, *m*, *c*, *k*; *rgb r, g, b*; or *ymck y, m, c, k*. Where, in the second and third formats "rgb:" and "ymck" are entered as text characters, followed by the numerical values. For example: *rgb 25, 35, 42*.

`* call SetLineWeight([obj], [val])`

Sets the line stroke weight (in points).

`* call SetLinePattern([pattern], [custpattern])`

Sets the current line pattern, or *pattern* to *custpattern*. *Custpattern* is specified in the form "onlength offlength onlength offlength onlength offlength," or as an integer from 0 to 9, representing one of the built in line patterns.

call **SetLineJoin**(jointype)

Sets the line join type. *jointype* can be: 0 (butt), 1 (bevel), 2 (rounded corner), 3 (miter).

pattern = **SetFillPattern**([obj], type, [[color1, [color2]], [steps], [centerX, centerY], [angle]])

Sets the fill pattern of the selected objects (or *obj*) to *type*, using *color1*, and *color 2*, with *steps* steps, starting at the center co-ordinates of *centerX*, *centerY*, and on an angle of *angle*. *Type* can be 0 through 3, representing the fill types: 0 is no fill, 1 is solid fill, 2 is a radial graduated fill, 3 is a linear graduated fill. Returns the currently selected line pattern or the line pattern of object *obj*.

call **SetGridOrder**(obj, Xlines, Ylines)

Sets the grid gridlines for *obj* to *Xlines* along the X axis, and *Ylines* along the Y axis.

objID = **GetObjNum**(obj)

Returns the object ID of *obj*.

objname = **GetObjName**(obj)

Returns the name of *obj*.

objsize = **GetObjSize**([obj])

Returns the size of the bounding box of object or group *obj*, (or the currently selected object or group if *obj* is not specified).

position = **GetObjPosn**([obj])

Returns the co-ordinates of the top left corner of the bounding box of object *obj*, (or the currently selected object if *obj* is not specified).

objsize = **GetObjVisSize**([obj])

Returns the size of the visible bounding box of object or group *obj*, (or the currently selected object or group if *obj* is not specified).

position = **GetObjVisPosn**([obj])

Returns the co-ordinates of the top left corner of the visible bounding box of object *obj*, (or the currently selected object if *obj* is not specified).

lock = **GetObjLock**([obj])

Returns the lock of object *obj*, (or the currently selected object if *obj* is not specified).

page = **GetObjPage**([obj])

Returns the page number on which object *obj*, (or the currently selected object if *obj* is not specified) is located.

`color = GetLineColor ([obj])`

Returns the current line color or the line color of object *obj*.

`num = GetLineWeight ([obj])`

Returns the currently selected line weight or the line weight of object *obj*.

`num = GetLinePattern ([obj])`

Returns the currently selected line pattern or the line pattern of object *obj*.

`num = GetLineJoin ([obj])`

Returns the currently selected line joint type or the line joint type of object *obj*.

`pattern = GetFillPattern ([obj], type, [[color1, [color2]], [steps], [centerX, centerY], [angle]])`

Gets the fill pattern of the selected objects (or *obj*), returning the *type*, *color 1*, *color 2*, *steps*, the center co-ordinates of *centerX*, *centerY*, and the *angle*. *Type* can be 0 through 3, representing the fill types: 0 is no fill, 1 is solid fill, 2 is a radial graduated fill, 3 is a linear graduated fill. Returns the currently selected line pattern or the line pattern of object *obj*.

`call GetGridOrder (obj)`

Gets the number of grid X and Y lines for *obj*.

`containpoint = OverlapPoint ([obj], X, Y)`

Returns whether or not the currently selected object (or object *obj*) contains the point X, Y.

`intersect = OverlapRect ([obj], left, top, right, bottom)`

Returns the intersection of the currently selected object (or object *obj*) and the rectangle *left*, *top*, *right*, *bottom*.

`bool = IsClosed (obj)`

Returns 1 if *obj* is a closed bezier curve. Returns 0 if it is not.

`bool = IsLocked (obj)`

Returns 1 if *obj* is a locked object. Returns 0 if it is not.

`bool = IsSelected (obj)`

Returns 1 if *obj* is currently selected. Returns 0 if it is not.

`bool = IsGrouped (obj)`

Returns 1 if *obj* is part a grouped object. Returns 0 if it is not.

`bool = IsCompound (obj)`

Returns 1 if *obj* is part of a compound object. Returns 0 if it is not.



`bool = IsBitmap(obj)`  
 Returns 1 if *obj* is a bitmap. Returns 0 if it is not.

`bool = IsEPSF(obj)`  
 Returns 1 if *obj* is an EPSF. Returns 0 if it is not.

`bool = IsStructGraphic(obj)`  
 Returns 1 if *obj* is a structured graphic. Returns 0 if it is not.

`bool = IsEllipse(obj)`  
 Returns 1 if *obj* is an ellipse. Returns 0 if it is not.

`bool = IsCircle(obj)`  
 Returns 1 if *obj* is a circle. Returns 0 if it is not.

`bool = IsBezier(obj)`  
 Returns 1 if *obj* is a bezier curve. Returns 0 if it is not.

`bool = IsGrid(obj)`  
 Returns 1 if *obj* is a grid. Returns 0 if it is not.

`bool = IsText(obj)`  
 Returns 1 if *obj* is text. Returns 0 if it is not.

`bool = IsFirst(obj)`  
 Returns 1 if *obj* is the first object in a compound. Returns 0 if it is not.

`bool = IsLast(obj)`  
 Returns 1 if *obj* is the last object in a compound. Returns 0 if it is not.

## Clip Operations

`call DefineClip(name)`  
 Defines the currently selected objects as a clip named *name*.

\* `call DrawClip(name, [X, Y])`  
 Places clip *name* at position *X*, *Y*.

`call DeleteClip(name)`  
 Deletes the clip *name* from the list.

\* `bool = LoadClips([filename])`  
 Loads the clip file *filename*. Returns 1 if the load is successful.

\* `bool = SaveClips([filename])`  
 Saves the clip list in a file named *filename*. Returns 1 if the save is successful.

`loadedclips = GetClipList()`  
 Returns a list of all the clips currently loaded.

## Environment Operations

\* call **SetTools()**

Brings up the tools requester.

call **SetUnits**(units)

*Units* can be 1 (inches), 2 (centimeters), 3 (Pica).

call **SetGridSize**(Xsize, Ysize)

Sets the resolution of the grid.

call **SetGrid**(status)

Clears or sets (0/1) the grid.

call **SetGridSnap**(status)

Clears/sets (0/1) snap-to-grid.

call **SetRulerType**(type)

Sets the ruler display units. *Type* can be set to: 1 (1/8"), 2 (1/6"), 3 (centimeters), or 4 (Picas).

call **SetRuler**(status)

Turns ruler display off when *status* is 0. Turns ruler display on when *status* is 1.

gsize = **GetGridSize**()

Returns grid resolution (x,y).

measure = **GetUnits**()

Returns type of units in use.

gstatus = **GetGrid**()

Returns grid status.

snap = **GetGridSnap**()

Returns snap-to-grid status.

measure = **GetRulerType**()

Returns ruler type. Type is 1, 2, 3, or 4 corresponding to 1/8", 1/6", CM, PICA.

rstatus = **GetRuler**()

Returns ruler status.

qm = **SetQuickMove**(status)

Clears/sets (0/1) Quick Move mode. This function returns the status that has been set.

wf = **SetWireframe**(status)

Clears/sets (0/1) Wireframe Graphics mode. This function returns the status that has been set.

`cm = SetColorMode(status)`

Sets Color mode to 0 or 1, corresponding to Color, B&W. This function returns the status that has been set.

`il = SetInterlace(status)`

Sets interlace on/off (0/1). This function returns the status that has been set.

`dm = SetDitherMode(mode)`

*Mode* is one of 0/1/2 corresponding to Smooth, Non- interlaced, Flicker Free.

`qm = GetQuickMove()`

Returns the current status of the Quick Move option.

`wf = GetWireframe()`

Returns the current status of the Wireframe option.

`cm = GetColorMode()`

Returns the current status of the *Professional Draw* screen's color mode.

`il = GetInterlace()`

Returns the current status of the *Professional Draw* screen's Interlace option.

`dm = GetDitherMode()`

Returns the current color dithering mode.

## Color Palette Operations

call **DefineColor**(*colorname*, R, G, B, Y, M, C, K)

Adds the color, *colorname* to the palette. The *RGB* values for *colorname* are specified as an integer from 0 to 15. *YMCK* values are specified as real numbers from 0.0 to 1.0.

call **TintColor**(*colorname*, *tint*)

Adds a tint of *colorname*, with a saturation of *tint*, to the palette.

call **AddPantone**(*pantonenumber*)

Adds the PANTONE Color named *pantonenumber*, to the palette.

`clrinfo = GetColorData(colorname)`

Returns the the component and flag information for *colorname* in the form: red, green, blue, yellow, magenta, cyan, black flags.

`clrinfo = GetPantoneData(pantonenumber)`

Returns the the component and flag information for *pantonenumber* in the form: red, green, blue, yellow, magenta, cyan, black, flags. The specified PANTONE Color does not have to be in the current palette.

```
clrs = GetColorList()
```

Returns a list of all the color names in the palette.

```
* call SaveColorList ([filename])
```

Saves the color data in a file called *filename*.

```
* call LoadColorList ([filename])
```

Loads the color data from the file called *filename*.

## Drawing Operations

All of the plotting commands in this section will accept a list of points separated by commas, in the form specified for that command.

```
call InitPlot (centerX, centerY, scaleX, scaleY, rotate,  
[bezname])
```

Initialized the plotting of a bezier object called *bezname*. This object centers around *centerX*, *centerY*, and is scaled by *scaleX*, *scaleY*, and rotated by *rotate* degrees. Subsequent point co-ordinates that are specified in plot commands will be shifted, scaled, and rotated accordingly.

```
call InitDraw (centerX, centerY, scaleX, scaleY, rotate,  
[objname])
```

Initialized the plotting of an object called *objname*. This object centers around *centerX*, *centerY*, and is scaled by *scaleX*, *scaleY*, and rotated by *rotate* degrees.

```
call AppendPlot (centerX, centerY, scaleX, scaleY, rotate,  
objname)
```

Initialized the plotting of a bezier object appended to *objname*. This object centers around *centerX*, *centerY*, and is scaled by *scaleX*, *scaleY*, and rotated by *rotate* degrees.

```
call AbortPlot ()
```

Aborts the current object being plotted.

```
lastpoint = PlotBezier (points)
```

Plots bezier *points*. Points should be specified in the form point, point, where point is: controlX, controlY, [tangentX, tangentY], [tangentX, tangentY]. Returns the X and Y co-ordinates of the last point plotted.

```
lastpoint = PlotLine (points)
```

Plots bezier *points*. Points should be specified in the form: point, point where point is: controlX, controlY. Tangents are calculated to create straight lines. Returns the X and Y co-ordinates of the last point plotted.

`lastpoint = PlotSmooth(points)`

Plots bezier *points*. Points should be specified in the form: controlX, controlY.

Tangents are calculated to create smooth curves. Returns the X and Y co-ordinates of the last point plotted.

`location = PlotTurtle(points)`

Plots bezier points. *Points* should be specified in the form point, point, where point is: angle, distance.

`location = MoveTurtle(points)`

Moves the cursor without plotting. *Points* should be specified in the form point, point, where point is: angle, distance.

`objID = EndPlot()`

Tells **Professional Draw** that the last point on the object has been plotted. Returns the new object's ID.

`objID = ClosePlot()`

Tells **Professional Draw** that the last point on the object has been plotted and closes the object in the style of the last draw command (bezier, smooth, line etc.). Returns the new object's ID.

`objID = DrawEllipse(Xpoint, Ypoint, width, height)`

Draws an ellipse centered at *Xpoint, Ypoint*, with diameters of *width, height*. Returns the object ID.

`objID = DrawRectangle(left, top, right, bottom, )`

Draws a rectangle with opposing corners at the *left, top*, and *right, bottom*. Returns the object ID.

`objID = DrawGrid(left, top, right, bottom, )`

Draws a grid with opposing corners at the *left, top*, and *right, bottom*. Returns the object ID.

`objID = EllipseToGraphic(ellipse)`

Converts *ellipse* to a bezier curve and returns the object ID.

## Text Operations

call `InitText([font, points, [setsize], [boldX, boldY], shear])`

Initializes the rendering of text with font *font*, point size *setsize*, bolding of *boldX* and *boldY*, and shear angle of *shear*.

call **Text**(coordX, coordY, text)

Renders *text* starting at the co-ordinates *coordX*, *coordY*.

call **AlignTextWithCurve**(justify, scale, rotate, reverse, standoff)

Aligns selected text with a curve. *justify*, *scale* *rotate* and *reverse* can each be set to either 0 or 1. 0 means do not justify, scale rotate or reverse. 1 means do justify, scale rotate or reverse.

call **TextToGraphic**()

Converts selected text to bezier curves.

fontsavail = **GetFontList**()

returns the list of fonts available fonts.

## Control Point Manipulation

numctrlpnts = **GetObjectOrder**(obj)

Returns the number of control points in *obj*.

pointinfo = **GetPoint**(obj, point)

Returns information about the control point *point* on *obj*. Information appears in the form: controlX, controlY, [tangentX, tangentY],[tangentX, tangentY].

call **SetPoint**(obj, ctrlpoint, newpoint)

Alters *ctrlpoint* in *obj* to reflect *newpoint*. Points should be specified in the form point, point, where point is: controlX,controlY, [tangentX, tangentY], [tangentX, tangentY]. Returns the X and Y co-ordinates of the last point plotted.

call **MovePoint**(obj, ctrlpnt, X, Y)

Moves the control point *ctrlpnt* in *obj* delta *X*,*Y*.

call **RotatePoint**(obj, ctrlpnt, angle)

Rotates the control point *ctrlpnt* in *obj* through *angle*.

call **ScalePointRadius**(obj, ctrlpnt, scale)

Scales the length of the tangents of control point *ctrlpnt* by *scale*.

call **SetPointRadius**(obj, ctrlpnt, lengthA, lengthB)

Sets the length of the tangents of control point *ctrlpnt* on *obj* to *lengthA* and *lengthB* respectively.

call **SmoothPoint**(obj, ctrlpnt, radius)

Sets the length of tangents for *ctrlpnt* on *obj*, to be continuous (co-linear) with speed *radius*. Note if *radius* is 0.0 then the curve around the point will be pointy, if *radius* is 1.0 the curve will be smooth, and if the *radius* is greater than 1.0 a "loopy" curve will result.

call **RemovePoint** (obj, ctrlpnt)

Deletes *ctrlpnt* from *obj*.

*newobjID* = **CutAtPoint** (obj, ctrlpnt)

Cuts *obj* at *ctrlpnt* and returns the new object ID if a new object is created.

call **CloseObject** (obj)

Closes the endpoints of *obj*. If the endpoints are not coincident, then a new point is generated.

call **OpenObject** (obj)

Opens *obj*. The same as **CutAtPoint**(obj, 0).

## Page PostScript Output Specs

These functions set and get page parameters which are accessed through the *Output Page Specifications* requester which appears when you click the *Postscript Output Specs* button in the *New Page Format* requester.

*pspsize* = **GetPSPageSize** ()

Returns the output page size.

call **SetPSPageSize** (Xsize, Ysize)

Sets the PostScript page dimensions (output page size).

*angle* = **GetPSOutputAngle** ([pgnm])

Returns the output angle of the specified page or the current page.

call **SetPSOutputAngle** (pgnm, angle)

Sets the output angle of the specified page.

*cropdata* = **GetPSOutputCrop** ([pgnm])

For the current or specified page, returns the status of crop marks (off or on), their length, open space and bleed.

call **SetPSOutputCrop** (pgnm, status (0 or 1), len, open, bleed)

For the specified page, sets whether crop marks are off or on (*status*), the crop mark lengths, open space and bleed.

*ejectstat* = **GetPSOutputEject** ([pgnm])

Returns the eject status (0 or 1) of the current page or the specified page.

call **SetPSOutputEject** (pgnm, eject)

For the specified page, sets whether eject is off (0) or on (1).

```
orient = GetPSOutputOrient ([pgnm])
```

Returns the orientation (0/1/2/3 corresponding to none, portrait, landscape or center) for the specified page or the current page.

```
call SetPSOutputOrient (pgnm, orient)
```

Sets the orientation (see above) for the specified page.

```
pospage = GetPSOutputPosn ([pgnm])
```

Returns the offset of PS output for the specified page or the current page.

```
call SetPSOutputPosn (pgnm, Xpos, Ypos)
```

Sets the offset of PostScript output for the specified page.

```
psscale = GetPSOutputScale ([pgnm])
```

Returns the output scale of the specified page or the current page.

```
call SetPSOutputScale (pgnm, Xsize, Ysize)
```

Sets the output scale of the specified page.

## PostScript Printing

The parameters for these functions refer to items found in the *Print to PostScript* requester.

```
csinksdata = GetCSInks ()
```

Returns data as with SetCSInks() for Process colours only.

```
call SetCSInks (k, y, m, c, [mech])
```

Indicates whether or not (1/0) to print the corresponding ink on color separation output. Mechanical inks are either all on or all off.

```
csdata = GetCSParams ()
```

Returns data as with SetCSParams() for Process colours only.

```
call SetCSParams (KLPI, KAng, YLPI, YAng, MLPI, MAng, CLPI, CAng, [MechAng], [MechLPI])
```

Sets colour separation densities and angles for black, yellow, magenta and cyan; optionally for mechanical colors. *MechAng* and *MechLPI* will apply to all mechanical colors.

```
auto = GetPSAutotile ()
```

Returns the off/on (0/1) status of PostScript autotiling.

```
call SetPSAutotile (status)
```

Turns PostScript autotiling off/on (0/1).



`status = GetPS8BitBM()`

Returns 0 or 1 corresponding to the off/on status of 8-bit bitmaps.

call `SetPS8BitBM(status)`

Sets 8-bit bitmaps off/on (0/1).

`epsf = GetPSEPSF()`

Returns the off/on (0/1) status of EPSF mode.

call `SetPSEPSF(status)`

Sets EPSF format off/on (0/1).

`fontdown = GetPSFontDownload()`

Returns 0 or 1 corresponding to font downloading option off/on.

call `SetPSFontDownload(status)`

Disables/enables (0/1) font downloading.

`manual = GetPSManFeed()`

Returns the off/on (0/1) status of manual feed.

call `SetPSManFeed(status)`

Sets manual feed off/on (0/1).

`mirror = GetPSMirror()`

Returns 0 or 1 corresponding to the off/on status of mirror printing.

call `SetPSMirror(status)`

Sets mirror printing off/on (0/1).

`neg = GetPSNegative()`

Returns 0 or 1 corresponding to the off/on status of negative printing.

call `SetPSNegative(status)`

Sets negative printing off/on (0/1).

`psover = GetPSOverride()`

Returns 0 or 1 corresponding to the off/on status of *Override Custom Specs*.

call `SetPSOverride(status)`

Turns *Override Custom Specs* off/on (0/1).

`device = GetPSOutput()`

Returns current PostScript output path; "PAR:", "SER:" or a filename.

call `SetPSOutput(spec)`

Sets PostScript output to be directed to *spec*; this is usually "SER:", "PAR:" or a specified filename.

`modetype = GetPSPrintMode()`

Returns the PostScript print mode which is one of 1/2/3/4 corresponding to B&W, Color PostScript, 3 Color and 4 Color.

`call SetPSPrintMode(mode)`

Sets the PostScript print mode as described above.

`proofmode = GetPSProof()`

Returns one of 0/1/2 corresponding to the modes of Draft, Proof and Final.

`call SetPSProof(mode)`

Sets the PostScript proof mode to one of 0/1/2 corresponding to Draft, Proof, and Final.

`rolldata = GetPSRollPaper()`

Returns the status of the roll paper option and, if on, the width of the paper.

`call SetPSRollPaper(status, [width])`

Sets RollPaper option off/on (0/1) and, optionally, the width of the roll.

`csdata = GetUCRGCR()`

Returns the UCR threshold and GCR values as percentages.

`call SetUCRGCR(ucrthres, gcr)`

Sets the UCR threshold value and the GCR value. Both arguments are expressed as a percentage.

`success = PrintDocPS([copies], [sync])`

Prints one or *copies* copies of the entire document. Returns 1 for success, 0 for an error or premature abort (caused by the user clicking on the *STOP!* button or by issuing an *ARexx AbortPrint* command). If *sync* is 1, the function returns only after printing is complete. If *sync* is 0 for asynchronous printing, the function will return immediately.

`success = PrintPagePS(pgnm, [copies], [sync])`

Prints one or *copies* copies of the current page. Returns 1 for success, 0 for an error or premature abort (as described above). If *sync* is 1, the function returns only after printing is complete. If *sync* is 0 for asynchronous printing, the function will return immediately.

## Thumbnail printing

These functions correspond to items in the *Print to Thumbnail* requester.

`thumbsize = GetThumbNum()`

Returns one of 2/3/4/5 corresponding to the size of Thumbnail printing; i.e., 2x2, 3x3, 4x4 and 5x5.

call `SetThumbNum(num)`

Sets the size of Thumbnail printing. *Num* is one of 2/3/4/5 corresponding to 2x2, 3x3, 4x4 and 5x5.

`success = PrintDocThumb([copies], [sync])`

Prints one or *copies* copies of the entire document. If *sync* is 1, the function returns only after printing is complete. If *sync* is 0 for asynchronous printing, the function will return immediately.



# 7. The Genie Editor

*Professional Draw* includes a simple text editor so that you can easily create and edit Genies within *Professional Draw*.

The Genie Editor has 3 menus, a *Project* menu an *Edit* menu and a *Find* menu, as well as a *Close* gadget. Each of the menu commands has a corresponding keyboard shortcut.

The Editor displays the current line number on the left hand side of the title bar.

## To Use the Genie Editor

New Function Genies may be created by clicking on the *Define* button in the Function Genie Requester. This will bring the *Genie Editor* to the front so that you may enter the *ARexx* script for your Function Genie. You may also edit existing scripts by selecting a Genie in the Function Genie Requester and clicking on the *Modify* button.

## Entering Text

Entering and editing text is very straightforward. You type text from the keyboard, and use the backspace and *DEL* keys to make corrections. You can move the cursor around using the cursor keys, or by clicking at a new position using the mouse.

## Project Menu

The *Project* menu contains the *Load*, *Save*, *Return* and *Quit* commands. This allows you to load and save Genies, or return to the *Professional Draw* window.

## Load (Amiga-O)

This command brings up the standard *Professional Draw* file requester. In this requester you select the name of the Genie you wish to edit, and click on *OK*. The text of the Genie will appear in the editor.

## Save (Amiga-S)

This command brings up the standard *Professional Draw* file requester. In this requester you enter the name and location where you want to save the Genie, and then click on *OK*. The file name will appear in the title bar of the Editor. By default, the file will be saved in the *rex:* directory, with the extension *.pdrx*.

## Return (Amiga-/)

This command returns you to the main *Professional Draw* window. The program will prompt you for a name for the newly created Genie, and it will automatically be saved in the *rex:* directory with the extension *.pdrx*. You can also exit the editor by clicking on the *Close* gadget.

## Quit (Amiga-Q)

This command returns you to the main *Professional Draw* window without saving any changes.

## The Edit Menu

The *Edit* menu contains the *Cut*, *Copy*, *Paste*, *Find*, and *Replace* commands. These commands give you the editing ability you need to create Genies.

### Cut (Amiga- X)

This command cuts the current line to the paste buffer, removing it from the file. The line can be replaced, or placed elsewhere with the *Paste* command. To cut a range of lines, simply repeat the cut command a number of times. Each *Cut* command will add a line to the information stored in the Paste buffer. You cannot cut a portion of a line.

### Copy (Amiga-C)

This command copies the current line to the paste buffer, leaving it in its current position in the file. The line can be placed elsewhere with the *Paste* command. To copy a range of lines, simply repeat the copy command a number of times. Each *Copy* command will add a line to the information stored in the Paste buffer. You cannot copy a portion of a line.

### Paste (Amiga-P)

This command pastes the information currently in the paste buffer at the current line.

### Go To Line... (Amiga-G)

This command brings up a requester that allows you specify what line you would like to jump to. Enter the line number in this requester, and click on *OK*. The cursor will jump to this line.

## The Find Menu

The *Find* menu contains the *Find*, *Find Next*, and *Replace* commands that allow you to find a character string that you are looking for, and replace it with another string if you wish.

## Find (Amiga-F)

This command brings up a requester asking you for the text you wish to find. Type the text you want to find. Select whether you want to search *Forward* from your current position in the document, *Backward* from your current position, or *From the Top* of the document. You can also specify whether you want to search for the exact string, including matching upper and lower case, or ignore the case in the search. Enabling *Upper/Lower Case* will cause the search to only find the exact string you have entered. When you are done, click on *OK*. Your current file will be searched, and the cursor will move to the text you were searching for. If the text is not found, a message will be displayed telling you so.

## Find Next (Amiga-N)

This command will find the next occurrence of the last string you searched for using the *Find* or *Replace* command.

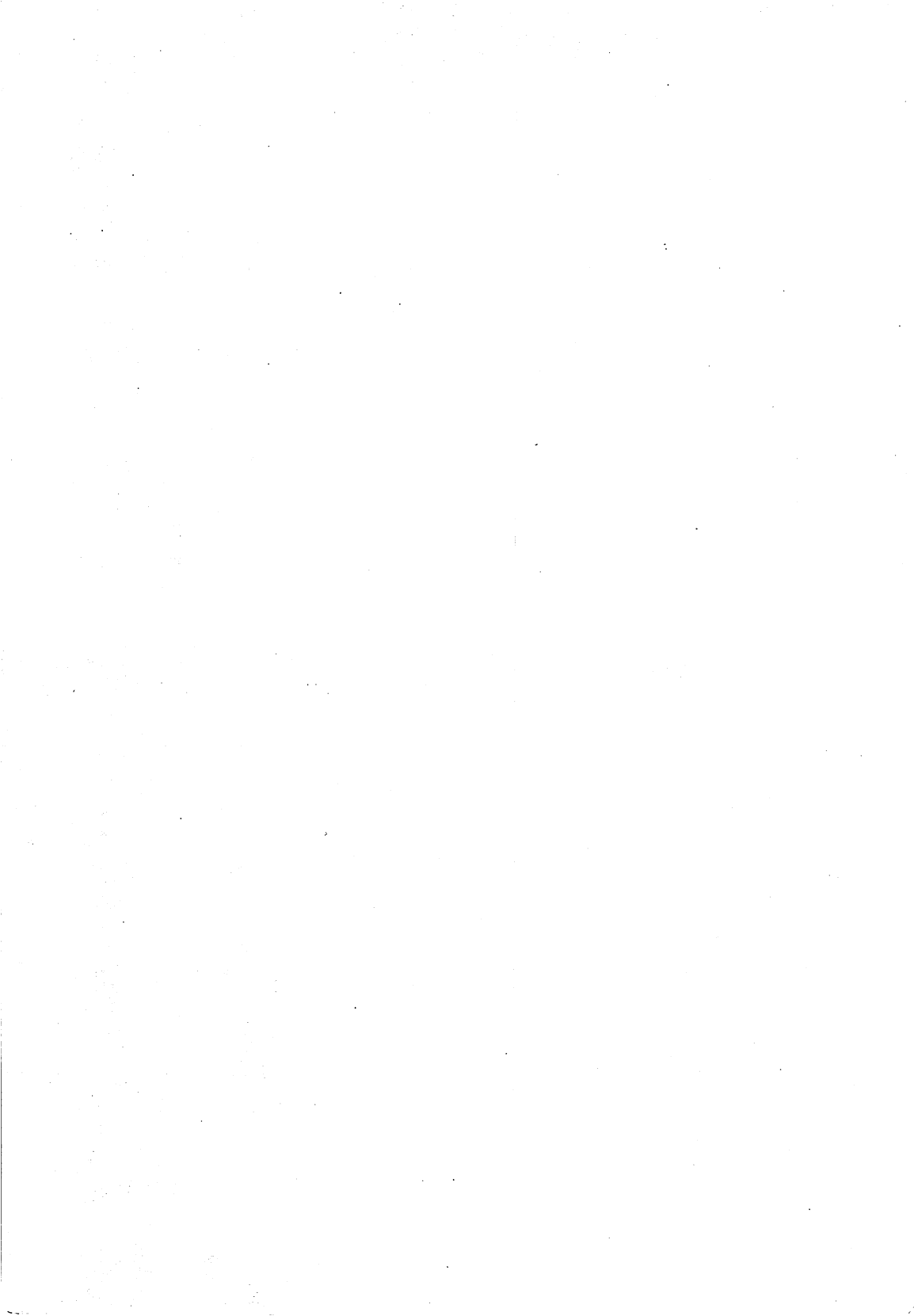
## Replace (Amiga-R)

This command brings up a requester asking you for the text you wish to find and replace. Type the text you want to change in the first text gadget, and the text you want to replace it with in the second text gadget. Select whether you want to search *Forward* from your current position in the document, *Backward* from your current position, or *From the Top* of the document. Specify whether you want all occurrences of the string changed automatically, by enabling *Replace All* or whether you want to be just change the next occurrence, by selecting *Replace Next*.

You can also specify whether you want to search for the exact string, including matching upper and lower case, or ignore the case in the search. Enabling *Upper/Lower Case* will cause the search to only find the exact string you have entered. Click on *OK* when you are finished. Your current file will be searched, and the cursor will move to the text you were searching for. If the text is not found, a message will be displayed telling you so.







Logfile:

SupraDrive-0: PDraw / install-log file





*GOLD DISK*